10-16-01 #2

500.40174X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): KASHIWAGI, Yugo

Serial No.: (Not yet assigned)

Filed: May 31, 2001

Title: SYSTEM DEVELOPING METHOD, STORAGE MEDIUM, INFORMATION PROCESSING APPARATUS, INFORMATION TERMINAL APPARATUS, INFORMATION PROCESSING SYSTEM, AND INFORMATION PROCESSING METHOD

LETTER CLAIMING RIGHT OF PRIORITY

Honorable Commissioner of
Patents and Trademarks
Washington, D.C. 20231

May 31, 2001

Sir:

Under the provisions of 35 USC 119 and 37 CFR 1.55, the applicant(s) hereby claim(s) the right of priority based on Japanese Patent Application No.(s) 2000-166468, filed May 31, 2000.

A certified copy of said Japanese Application is attached.

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP

Carl I. Brundidge
Registration No. 29,621

CIB/alb
Attachment
(703)312-6600

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されてい る事項と同一であることを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２０００年　５月３１日

出 願 番 号
Application Number: 特願２０００－１６６４６８

出 願 人
Applicant (s): 株式会社日立製作所

２００１年　４月１３日

特 許 庁 長 官
Commissioner,
Patent Office 及 川 耕 造

【書類名】　特許願

【整理番号】　H00002961

【提出日】　平成12年 5月31日

【あて先】　特許庁長官殿

【国際特許分類】　G06F　9/44

【発明者】

【住所又は居所】　東京都小平市上水本町五丁目２０番１号　株式会社日立製作所　半導体グループ内

【氏名】　柏木　有吾

【特許出願人】

【識別番号】　000005108

【氏名又は名称】　株式会社日立製作所

【代理人】

【識別番号】　100089071

【弁理士】

【氏名又は名称】　玉村　静世

【電話番号】　047-361-8861

【手数料の表示】

【予納台帳番号】　011040

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　システム開発方法、記憶媒体、情報処理装置、情報端末装置、情報処理システム及び情報処理方法

【特許請求の範囲】

【請求項１】　複数の記述を有するソースプログラムにおける第１の記述を、第１の計算機で実行可能な第１コードに変換し、

前記第１の記述を、該第１の記述によって定義される機能とは異なる機能を表すところのコードであって、第２の計算機で実行可能な第２コードに変換すると共に、前記ソースプログラムにおける第２の記述を、前記第２の計算機で実行可能なその他のコードに変換し、

前記ソースプログラムのデバッグにおいて前記第２の計算機による前記第２コードの実行に応答して前記第１の計算機が前記第１コードを実行することを特徴とするシステム開発方法。

【請求項２】　前記第１の計算機は、前記第１コードを実行する前に前記第２計算機から情報を取り込むことを特徴とする請求項１記載のシステム開発方法。

【請求項３】　前記第２コードは割り込み命令を表すコードであり、該コードの実行により前記第１の計算機へ、前記第２コードのアドレス情報を供給することを特徴とする請求項２記載のシステム開発方法。

【請求項４】　前記第２コードのアドレス情報と前記第１コードとを対応させる対照データを第１の計算機が参照可能に生成することを特徴とする請求項１記載のシステム開発方法。

【請求項５】　コンパイラを用いて前記第１コードへの変換、第２コードへの変換、及びその他のコードへの変換を行うことを特徴とする請求項１乃至４の何れか１項記載のシステム開発方法。

【請求項６】　デバッガを用いて前記ソースプログラムのデバッグを行なうことを特徴とする請求項５記載のシステム開発方法。

【請求項７】　前記第１の計算機はデバッガシステムを構成する計算機であり、前記第２の計算機はターゲット装置を構成する計算機であることを特徴とす

る請求項１乃至４の何れか１項記載のシステム開発方法。

【請求項８】　前記第１の計算機はホスト装置を構成する計算機であり、前記第２の計算機はターゲット装置である端末装置を構成する計算機であることを特徴とする請求項１乃至４の何れか１項記載のシステム開発方法。

【請求項９】　第１の記述と第２の記述を含むソースプログラムを入力する入力ステップと、入力した第１の記述を第１の計算機によって実行可能なコードに変換する第１変換ステップと、入力した前記第１の記述を該第１の記述により表される機能とは異なる機能であって、第２の計算機によって実行可能なコードへ変換する第２変換ステップと、入力した第２の記述を前記第２の計算機によって実行可能なコードに変換する第３変換ステップとを実行するプログラムを計算機によって読み取り可能に記憶して成るものであることを特徴とする記憶媒体。

【請求項１０】　前記プログラムは、１個の記憶媒体に格納されていることを特徴とする請求項９記載の記憶媒体。

【請求項１１】　前記プログラムは、伝送線を介して記憶媒体に取り込まれることを特徴とする請求項９記載の記憶媒体。

【請求項１２】　前記第２の計算機によって実行可能なコードは、割り込みを表す割り込みコードであることを特徴とする請求項９乃至１１の何れか１項記載の記憶媒体。

【請求項１３】　前記プログラムは更に、前記割り込みコードにリンクされる割り込みハンドラを規定するステップを含むことを特徴とする請求項１２記載の記憶媒体。

【請求項１４】　前記割り込みコードは、その実行により、前記第２の計算機のオペレーティングシステムが規定する割り込みハンドラの処理に分岐するものであることを特徴とする請求項１２記載の記憶媒体。

【請求項１５】　前記プログラムは更に、前記第２の計算機における前記割り込みコードのアドレスと前記第１の計算機が実行可能なコードとの対応を示すテーブルを形成するステップを含むことを特徴とする請求項１２記載の記憶媒体。

【請求項１６】　前記第１の計算機はデバッガシステムを構成する計算機であり、前記第２の計算機はターゲット装置を構成する計算機であることを特徴とする請求項９乃至１７の何れか１項記載の記憶媒体。

【請求項１７】　前記第１の計算機はホスト装置を構成する計算機であり、前記第２の計算機は情報端末装置を構成する計算機であることを特徴とする請求項９乃至１５の何れか１項記載の記憶媒体。

【請求項１８】　中央処理装置を含む情報端末装置から当該情報端末装置で実行されるプログラムにおける所定の命令コードのアドレス情報を受け取り、受け取ったアドレス情報に基づいて前記情報端末装置の内部情報を取り込み、取り込んだ内部情報を用いて、そのアドレス情報に対応して規定されている処理を実行するものであることを特徴とする情報処理装置。

【請求項１９】　前記取り込んだアドレス情報に基づいて、取り込むべき前記情報のアドレスを求めることを特徴とする請求項１８記載の情報処理装置。

【請求項２０】　前記所定の命令コードは割り込み命令を表すコードであることを特徴とする請求項１９記載の情報処理装置。

【請求項２１】　特定の命令コードを含むプログラムを格納し、中央処理装置による前記特定の命令コードの実行によって、当該命令コードのアドレスを外部へ送信することを特徴とする情報端末装置。

【請求項２２】　前記特定の命令コードは、割り込み命令を表す命令コードであることを特徴とする請求項２１記載の情報端末装置。

【請求項２３】　前記アドレスの外部送信に応答して外部から供給される外部要求を入力し、入力した外部要求に応答して所定の内部情報を外部へ提供することを特徴とする請求項２１又は２２記載の情報端末装置。

【請求項２４】　伝送路に接続可能なホスト装置を有する情報処理システムであって、前記ホスト装置は、アドレス情報に対応した機能を実行可能な実行ユニットを有し、プログラムを格納した情報端末装置から出力されるところの前記プログラムのメモリ空間におけるアドレス情報を受信し、受信したアドレス情報に対応した機能を提供するものであることを特徴とする情報処理システム。

【請求項２５】　前記実行ユニットは、複数のアドレス情報にそれぞれ対応した機能を提供可能とされていることを特徴とする請求項２４記載の情報処理システム。

【請求項２６】　前記実行ユニットは、受信したアドレス情報に基づいて前記情報端末装置へのアクセス情報を形成することを特徴とする請求項２５記載の情報処理システム。

【請求項２７】　前記プログラムには、割り込み命令を表すコードが設けられ、該割り込み命令を表すコードの実行により、前記アドレス情報を送信することを特徴とする請求項２４乃至２６の何れか１項記載の情報処理システム。

【請求項２８】伝送路に接続可能なホスト装置を有する情報処理システムであって、前記ホスト装置は、互いに関連する第１のプログラムと第２のプログラムの内、第１のプログラムを端末装置へ提供し、該端末装置からの要求に応答して、前記第２のプログラムを実行することを特徴とする情報処理システム。

【請求項２９】　前記第１のプログラムは、前記情報処理システムから前記情報端末装置へ送信されることを特徴とする請求項２８記載の情報処理システム。

【請求項３０】　前記第１のプログラムと前記第２のプログラムは、共通のソースプログラムより形成されるものであることを特徴とする請求項２８又は２９記載の情報処理システム。

【請求項３１】　前記要求は、前記第１のプログラムにおけるアドレス情報を含んでいることを特徴とする請求項２８乃至３０の何れか１項記載の情報処理システム。

【請求項３２】　情報端末装置に接続可能なホスト装置を有する情報処理システムであって、前記ホスト装置は、第１の処理をホスト装置へ要求するところのプログラムを情報端末装置へ提供し、情報端末装置からの要求に応じて前記第１の処理を実行することを特徴とする情報処理システム。

【請求項３３】　前記プログラムは、伝送路へ向けて提供されることを特徴とする請求項３２記載の情報処理システム。

【請求項３４】　前記要求は伝送路から供給されることを特徴とする請求項３２又は３３記載の情報処理システム。

【請求項３５】　前記情報端末装置へ提供されるべき前記プログラムと、前記第１の処理に対応するプログラムとは、共通のソースプログラムより形成されたものであることを特徴とする請求項３４記載の情報処理システム。

【請求項３６】　ネットワークを介してサーバーがクライアントにサービスを提供する情報処理方法であって、サーバーは、プログラムを格納した情報端末装置から出力されるところの前記プログラムのアドレス空間におけるアドレス情報をネットワークを介して受信し、受信したアドレス情報に対応するサービスの要求に応答するための処理を実行ユニットで実行し、実行結果に基づくサービスをネットワークを介してサーバーに送信することを特徴とする情報処理方法。

【請求項３７】　前記実行ユニットは、複数のアドレス情報にそれぞれ対応する処理の中から、受信したアドレス情報に対応する処理を選択して、その処理を実行することを特徴とする請求項３６記載の情報処理方法。

【請求項３８】　前記実行ユニットによる処理の実行は、受信したアドレス情報に基づいて前記クライアントから読み込むべき情報をアクセスするためのアクセス情報の生成を含むことを特徴とする請求項３７記載の情報処理方法。

【請求項３９】　前記プログラムには割り込み命令を表すコードが設けられ、該割り込み命令を表すコードの実行により、前記アドレス情報をネットワークに送信することを特徴とする請求項３６乃至３８の何れか１項記載の情報処理方法。

【請求項４０】　ネットワークを介してサーバーがクライアントにサービスを提供する情報処理方法であって、サーバーは、関連する第１のプログラムと第２のプログラムの内、第１のプログラムをネットワークを介してクライアントへ送信し、該クライアントからの要求に応答して、前記第２のプログラムを実行することを特徴とする情報処理方法。

【請求項４１】　前記第１のプログラムと前記第２のプログラムは、共通のソースプログラムより形成されるものであることを特徴とする請求項４０記載の情報処理方法。

【請求項４２】　前記要求は、前記第１のプログラムのアドレス空間におけるアドレス情報を含んでいることを特徴とする請求項４０又は４１記載の情報処理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明はＣ＋＋言語等の高級言語を用いて組込みシステムを開発するのに最適なシステム開発方法、そのためのプログラムを格納した記憶媒体、更には、ネットワークを介したサービスの提供を受けるクライアント若しくは情報端末装置における処理負担の軽減、更には、そのような負担軽減を考慮したサービス提供のための情報処理システム及び情報処理方法に関する。

【０００２】

【従来の技術】

計算機（本明細書においてコンピュータ装置などの電子計算機を単に計算機と称する）の開発言語であるＣ＋＋，Ｆｏｒｔｒａｎ９０、Ｊａｖａ等の高級言語が持つ高機能（例外処理など）は大きなリソース（スタック、メモリ等）を必要とする。このため、搭載メモリなどに制限の多い組込みシステム若しくは携帯情報端末装置等の開発言語にはＣ＋＋のような高級言語に対するサブセット言語が用いられている。例えばＥＣ＋＋というＣ＋＋のサブセット言語が提案されている。このようなサブセット言語を用いるときは、Ｃ＋＋が持つ例外処理機能等を含まないサブセットの範囲でソースプログラムを記述し、サブセット言語の処理系でコンパイルする。これにより、従来のＣ言語と同程度のリソース効率で高級言語仕様を実現できる。

【０００３】

【発明が解決しようとする課題】

しかしながら、サブセット言語ではプログラムの標準化が困難である。すなわち、パーソナルコンピュータやワークステーション上で開発したプログラムを組み込み用途のマイクロコンピュータ等に移植することが困難になる。また、サブセットでは、プログラムデバッグ中だけに使用したい言語機能（例えば入出力、

例外処理等）が完備していないため、フルセットの開発言語環境と同様のデバッグを行なうことができず、デバッグ効率が悪くなる、という問題点のあることが本発明者によって明らかにされた。

【０００４】

そこで本発明者はターゲット装置（例えば組み込み用途のマイクロコンピュータ等）にとってオーバーヘッドになり得る高級言語機能も含めてフルセットの高級言語機能をシステム開発に利用可能にするために、ターゲット装置に対する負荷の軽減を行なうことを検討した。特に、ここで検討している負荷の軽減は、マルチプロセッサ向けコンパイラのように、処理能力が均質若しくは同等のプロセッサを対象として処理を分散させるという技術、とは基本的な着眼点が相違する、ということは本発明者によって予め把握されている。本発明者による検討内容は高級言語機能単位で負荷の軽減を図ろうとするものである。

【０００５】

更に本発明者が検討している負荷の軽減とは、開発環境の整備、即ちサブセットの開発言語ではプログラムデバッグ中だけでも使用したい高級言語機能（例えば入出力、例外処理等）が完備していないからデバッグ効率も悪くなるという問題点の解決、を出発点としており、この意味で、ターゲット装置の負荷の軽減に対して、軽減されるべき処理をデバッガに担わせることを検討した。特に、デバッガはコンパイルの過程で生成されるデバッグ情報を利用することにより軽減されるべきターゲット装置の処理を容易に実現できる、という点に着目している。

【０００６】

本発明者、そのターゲット装置とデバッガシステムとの関係を、ターゲットシステムの開発環境だけでなく、実システムにも拡張する検討を加えた。特にホスト（サーバー）と端末装置（クライアント、ターゲット装置）とを備えるような情報処理システムでは端末装置のリソースに大きな制限のある場合が想定され、特に、端末装置がＰＤＡ（Personal Digital Assistants）のような携帯情報端末装置や携帯電話機では既にその傾向が著しい。このような情報処理システムにおいて端末装置の負荷を軽減することを考慮したサービス提供のための情報処理システム若しくは情報処理方法によるビジネスモデルについても本発明者は検討

した。

【０００７】

本発明の目的は、ターゲット装置にとってオーバーヘッドになり得る高級言語機能も含めてフルセットの高級言語機能を利用可能にするシステム開発方法を提供することにある。

【０００８】

本発明の別の目的はＣ＋＋言語等の高級言語を用いて組込みシステムを開発するのに最適なシステム開発方法を提供することにある

本発明の更に別の目的は、前記システム開発方法によるシステム開発を容易化することができるプログラムを格納した記憶媒体を提供することにある。

【０００９】

本発明のその他の目的は、ネットワーク等を介したサービスの提供を受ける情報端末装置の処理負担軽減を実現する情報処理装置、情報端末装置、並びに情報処理システムを提供することにある。

【００１０】

本発明のその他の目的は、ネットワーク等を介してクライアント若しくは情報端末装置にサービスを提供するデータ処理システムの低コスト化、及びクライアントとの間の通信の簡素化を実現することにある。

【００１１】

本発明の前記並びにその他の目的と新規な特徴は本明細書の記述及び添付図面から明らかになるであろう。

【００１２】

【課題を解決するための手段】

本願において開示される発明のうち代表的なものの概要を簡単に説明すれば下記の通りである。

【００１３】

〔１〕先ず基本的な考え方を一例として説明する。リソースなどの点で開発対象のターゲット装置がサポートできない若しくはターゲット装置にサポートさせたくない高級言語の高機能対応部分（例えばＣ＋＋の例外処理）が軽減対象になる

。ソースプログラムの内の当該高機能対応部分については、その機能をデバッガシステム（若しくはホスト装置）で実現する為の第１コードに変換すると共に、デバッガシステム（若しくはホスト装置）にその第１コードを処理させるための要求等を発生させるターゲット用の第２コードに変換する。上記変換は例えばコンパイラ等を利用して行なわれる。更に、ターゲット装置のメモリ及びレジスタをアクセス可能にするモニタ機能として、デバッガシステム（若しくはホスト）用のモニタコマンド列（クライアントアクセスコマンド列）を生成する。モニタコマンド列は第１コードと同様に、第２コードの実行に応答して起動される性質のコードである。

【００１４】

上記変換若しくはコンパイルされたオブジェクトプログラムをデバッガを用いてターゲット装置に実行させてソースプログラムの検証を行なうとき、ターゲット装置は高級言語の前記高機能対応部分では前記第２コードを実行する。第２コードは例えばＴＲＡＰ命令のような割り込み命令のコードであり、高級言語の前記高機能対応部分を実行したとするときのオーバヘッドはない。第２コードがＴＲＡＰ命令であればその命令コード固有のベクタ若しくは分岐先アドレスによって決まる割り込みハンドラを実行して、ターゲット装置はデバッガと通信を行ない、そのＴＲＡＰ命令のアドレス（ＰＣアドレス）をデバッガに与える。デバッガは、与えられたＴＲＡＰ命令のアドレスに応ずるターゲット上のスタック領域の情報などを取り込み、前記与えられたＴＲＡＰ命令のアドレスに対応する第１コードを、前記取り込んだスタック領域等の情報を用いて実行する。このようにして、ホスト装置が前記ターゲット装置のための前記高機能対応部分を実行若しくは模擬する。これにより、高級言語の機能単位に着目してターゲット装置の負荷を軽減することができる。デバッガによる前記スタック領域等に関する情報取り込みはモニタコマンド列を実行して行なう。モニタコマンド列によるアクセス機能は、前記割り込みハンドラの機能を介在させて実現してよい。

【００１５】

前記高級言語の高機能対応部分が専らプログラムデバッグ中だけに使用したい例外処理などの言語機能である場合、デバッグ完了後、最終のオブジェクトプロ

グラム中のＴＲＡＰ命令のような第２コードをＮＯＰに置き換えたりすればよい。この場合、ターゲット装置は、スタンドアロンシステムとしてホストから切り離して動作可能である。

【００１６】

最終プログラムに前記第２コードを残す必要がある場合、例えば実システムにおいてネットワーク上でクライアントの負荷を軽減することを想定するときは、最終プログラムにＴＲＡＰ命令のような第２コードを残し、ホスト装置には前記第１コードやモニタコマンド列等を組み込み、最終システム稼動後も、高級言語の高機能部分であるオーバーヘッド部分をネットワークのホスト装置に実行させることが可能になる。

【００１７】

〔２〕《システムの開発方法》先ず本発明をシステム開発方法の観点から具体的に説明する。

【００１８】

システム開発方法は、①一つの開発言語で記述され複数の記述を有するソースプログラム（４）における第１の記述を、第１の計算機（１）で実行可能な第１コードに変換し、②前記第１の記述を、該第１の記述によって定義される機能とは異なる機能を表すところのコードであって、第２の計算機（３）で実行可能な第２コード（ＴＲＡＰ命令コード）に変換すると共に、前記ソースプログラムにおける第２の記述を、前記第２の計算機で実行可能なその他のコード（第３コード）に変換し、③前記ソースプログラムのデバッグにおいて前記第２の計算機による前記第２コードの実行に応答して前記第１の計算機が前記第１コードを実行することを特徴とする。これにより、第２の計算機にとってオーバヘッドになると考えられる高級言語の高機能部分（第１の記述）単位で第１の計算機にその処理を負わせることが可能になる。要するに、第２の計算機にとってオーバヘッドになると考えられる高級言語の高機能部分（第１の記述）の処理を、第２計算機のリソースを用いることなく実現できる。

【００１９】

前記第１の計算機は、前記第１コードを実行する前に前記第２計算機から情報

を取り込む。例えば、第１の計算機は前記モニタコマンド列を実行し、第２のコードを実行したときの第２の計算機の内部状態、例えばスタック領域の状態等を取得する。第１の計算機は、前記スタック領域の情報等を参照し、第１コードを実行することにより、前記第２の計算機にとってオーバヘッドになると考えられる高級言語の高機能部分（第１の記述）を模擬する。

【００２０】

前記第２コードはＴＲＡＰのような割り込み命令を表すコードである。第２の計算機は、当該割り込みコードの実行により割り込みハンドラによる処理に分岐する。割り込みハンドラは、例えば前記第１の計算機へ、前記第２コードのアドレス情報、例えば前記割り込み命令の命令アドレスを供給する処理を規定する。割り込みハンドラは、例えば第２計算機のＯＳ（オペレーティングシステム）の一部に組み込まれ、或いはデバッグモニタプログラムとして第２のコードにリンクさせることができる。

【００２１】

前記第２コードのアドレス情報と前記第１コードとを対応させる対照データを第１の計算機で参照可能に生成しておく。これにより、第１計算機は、第２計算機による第２コードの実行に応答して実行すべき第１コードを複数の第１コードの中から簡単に選べる。

【００２２】

前記第１コードへの変換、第２コードへの変換、及びその他のコードへの変換はコンパイラを用いて行なえばよい。前記ソースプログラムのデバッグはデバッガを用いて行なえばよい。

【００２３】

開発されるべきシステムに着目したとき、ターゲット装置が単独の開発対象として位置付けられる場合、前記第１の計算機はデバッガシステムを構成する計算機であり、前記第２の計算機はターゲット装置を構成する計算機になる。このときの実システムにおいてソースプログラムの実行はターゲット装置単独で可能である。換言すれば、前記高級言語の高機能対応部分が専らプログラムデバッグ中だけに使用したい例外処理などの言語機能であて、実システムでは実行させる必

要がない場合、デバッグ完了後、最終のオブジェクトプログラム中のＴＲＡＰ命令のような第２コードを例えばＮＯＰに置き換えて、ターゲット装置は、スタンドアロンシステムとしてホスト装置から切り離して動作可能になる。

【００２４】

一方、実システムにおいて例えばネットワーク上でクライアントのようなターゲット装置の負荷を軽減することを想定するときは、最終プログラムにＴＲＡＰ命令のような第２コードを残し、ホスト装置には第１コード及びアクセスコマンド列を組み込み、最終システム稼動後も、高級言語の高機能部分であるオーバーヘッド部分をネットワークのホスト装置に実行させることが可能になる。このような場合、前記第１の計算機にはネットワークのホスト装置を構成する計算機を想定し、前記第２の計算機はターゲット装置である情報端末装置を構成する計算機を想定する。

【００２５】

〔３〕《プログラムの記憶媒体》前記システム開発方法の実現に利用されるプログラムを格納した記憶媒体の観点より本発明を説明する。

【００２６】

記憶媒体（３１，３２）は計算機によって読取り可能にプログラムを記録する。記録されたプログラムは、第１の記述と第２の記述を含むソースプログラムを入力する入力ステップ（Ｓ１）と、入力した第１の記述を第１の計算機によって実行可能なコードに変換する第１変換ステップ（Ｓ６）と、入力した前記第１の記述を該第１の記述により表される機能とは異なる機能であって、第２の計算機によって実行可能なコードへ変換する第２変換ステップ（Ｓ４）と、入力した第２の記述を前記第２の計算機によって実行可能なコードに変換する第３変換ステップ（Ｓ３）と、を実行させる。この記憶媒体を計算機に読取らせてコンパイラシステムやデバッガシステムを構成すると、前述のシステム開発方法を容易に実現することができる。

【００２７】

前記プログラムは、１個の記憶媒体に格納しても、或いは複数個の記憶媒体に分散させても、或いは別のデータやプログラムと一緒に記憶媒体に記録してもよ

い。

【００２８】

前記プログラムを記憶媒体に取り込む手段は何ら限定されず、ＣＤ－ＲＯＭなどの様にプラスチック成形により、ハードディスクやフロッピーディスクなどのように磁気的変化によって、或いは、伝送線を介して記憶媒体に取り込まれてよい。

【００２９】

前記第２の計算機によって実行可能なコードが、割り込みを表す割り込みコードであるなら、前記プログラムは、前記割り込みコードにリンクされる割り込みハンドラを規定するステップを更に含んでよい。その割り込みハンドラを前記第２の計算機のオペレーティングシステムがサポートする場合にはリンクされる割り込みハンドラの内容を一々規定しなくてよい。

【００３０】

前記プログラムは、前記第２の計算機における前記割り込みコードのアドレスと前記第１の計算機が実行可能なコードとの対応を示すテーブルを形成するステップを更に含んでよい。

【００３１】

前記プログラムを読み取って実行する計算機はコンパイラシステムやデバッガシステムを構成する計算機である。このとき、前記第１の計算機はデバッガシステムを構成する計算機であり、前記第２の計算機はターゲット装置を構成する計算機である。開発システムのネットワーク利用を予め想定しているとき、前記第１の計算機はホスト装置を構成する計算機であり、前記第２の計算機は情報端末装置を構成する計算機である。

【００３２】

〔４〕《ホスト装置》前記システム開発方法等によって開発されるターゲットのための前記高機能対応部分を実行若しくは模擬するホスト装置のような情報処理装置の観点より本発明を説明する。

【００３３】

情報処理装置（４０）は、中央処理装置（５０）を含む情報端末装置（４１）

から当該情報端末装置で実行されるプログラムにおける所定の命令コード例えば割り込み命令を表す命令コードのアドレス情報を受け取り、受け取ったアドレス情報に基づいて前記情報端末装置の内部情報を取り込み、取り込んだ内部情報を用いて、そのアドレス情報に対応して規定されている処理を実行する。この情報処理装置は、前記システム開発方法によって開発されるところの第２の計算機を有するターゲット装置を用いる実システムにおけるホスト装置に最適であり、ターゲット装置とされた情報端末装置におけるオーバヘッド機能を代替することができる。

【００３４】

情報端末装置は前記内部情報を取り込むために、前記取り込んだアドレス情報に基づいて、取り込むべき前記内部情報のアドレスを求めれば、端末装置に負担をかけずに済む。

【００３５】

〔５〕《情報端末装置》前記システム開発方法等によって開発されるターゲットの観点より本発明を説明する。

【００３６】

前記ターゲットのような情報端末装置（４１）は、割り込み命令を表す命令コードのような特定の命令コードを含むプログラムを格納し、中央処理装置（５０）による前記特定の命令コードの実行によって、当該命令コードのアドレスを外部へ送信する。前記アドレスの外部送信に応答して外部から供給される外部要求を入力し、入力した外部要求に応答して所定の内部情報を外部へ提供する。割り込み命令のような命令を実行するだけで、情報端末装置がそのプログラムメモリ空間上のどの命令実行状態にあるかをホスト装置に連絡でき、ホスト装置は、そのアドレスに対応する処理を行なうことによって、情報端末装置にとってオーバヘッドの大きな処理を代わって実行することができる。

【００３７】

〔６〕《情報処理システム》伝送路を介してクライアント若しくは情報端末装置にサービスを提供する情報処理システムの観点から本発明を説明する。

【００３８】

サービスの要求を伝送路若しくはその集合体であるネットを介して受け、そのサービスを提供するサーバーを有するような情報処理システムを想定したとき、クライアントからサーバーのプログラムを呼び出すのではなく、サーバーはクライアントの命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供する。即ち、伝送路に接続可能なホスト装置（４０）を有する情報処理システムにおいて、ホスト装置は、アドレス情報に対応した機能を実行可能な実行ユニット（５３Ｐ，５３Ｍ）を有し、プログラムを格納した情報端末装置（４１，４２，４３）から出力されるところの前記プログラムのメモリ空間におけるアドレス情報を受信し、受信したアドレス情報に対応した機能を情報端末装置に提供する。このように、ホスト装置（サーバー）は情報端末装置（クライアント）の命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供するというプロトコルを採用するから、ホスト装置と情報端末装置との間の通信処理が簡素であり、また、情報端末装置において前記アドレス情報を出力する処理を規定するプログラムとホスト装置においてアドレス情報に応じた処理を行なうプログラムを前述のシステム開発方法を適用して形成することができる。したがって、ネットワーク等を介してクライアント若しくは情報端末装置にサービスを提供するデータ処理システムの低コスト化も実現できる。

【００３９】

前記実行ユニットは、複数のアドレス情報にそれぞれ対応した機能を提供可能とされる。前記実行ユニットは、受信したアドレス情報に基づいて前記情報端末装置をアクセスするためのアクセス情報を形成したりしてよい。

【００４０】

前記プログラムには、割り込み命令を表すコードが設けられ、該割り込み命令を表すコードの実行により、前記アドレス情報を送信してよい。

【００４１】

〔７〕《情報処理システム》伝送路を介してプログラムを提供するような情報処理システムを想定する。即ち、インターネット等を介するプログラムのダウンロード、或いは、クライアント内で作られたプログラムにおいて使われるソフト部

品をサーバーが提供するようなシステムを想定する。このとき、クライアントのリソースの観点より、全てのプログラムを自分で実行することが困難若しくは不経済な場合がある。これに着目して、クライアントが欲するプログラムを、少なくとも２つに分け、サーバーは、一方をクライアントに提供し、クライアントからの要求で他方のプログラムを実行する。例えば、通信ネットワークに接続可能なホスト装置（４０）を有する情報処理システムであって、前記ホスト装置は、一体として使われるべき第１のプログラム（ＰＧＭ１）と第２のプログラム（ＰＧＭ２）の内、第１のプログラムを情報端末装置（４１，４２，４３２）へ提供し、該端末装置からの要求に応答して、前記第２のプログラムを実行する。

【００４２】

前記第１のプログラムは、前記情報処理システムから前記情報端末装置へ送信されてよい。前記第１のプログラムと前記第２のプログラムは、共通のソースプログラムより形成されてよい。この場合には前述のシステム開発方法を適用可能である。前記要求は、前記第１のプログラムにおけるアドレス情報を含んよい。

【００４３】

〔８〕《情報処理システム》クライアントによる処理のオーバヘッドを解決するという観点に立ち返れば、情報端末装置に接続可能なホスト装置（４０）を有する情報処理システムにおいて、前記ホスト装置は、第１の処理をホスト装置へ要求するところのプログラムを情報端末装置（４１，４２，４３）へ提供し、情報端末装置からの要求に応じて前記第１の処理を実行する。前記プログラムは、伝送路（４４）へ向けて提供され、前記要求は伝送路から供給される。システムのコスト低減のためには、前述の通り、情報端末装置へ提供されるべき前記プログラムと、前記第１の処理に対応するプログラムとを、共通のソースプログラムより形成するとよい。

【００４４】

〔９〕《情報処理方法》ネットワーク等を介してクライアント若しくは情報端末装置にサービスを提供する情報処理方法の観点に立って本発明を説明する。

【００４５】

前述の情報処理システムの観点と同様に、サービスの要求をネットを介して受

けることを想定したとき、クライアントからサーバーのプログラムを呼び出すのではなく、サーバーはクライアントの命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供する。即ち、ネットワークを介してサーバーがクライアントにサービスを提供する情報処理方法であって、サーバーは、プログラムを格納した情報端末装置から出力されるところの前記プログラムのアドレス空間におけるアドレス情報をネットワークを介して受信し、受信したアドレス情報に対応するサービスの要求に応答するための処理を実行ユニットで実行し、実行結果に基づくサービスをネットワークを介してサーバーに送信する。このように、サーバーはクライアントの命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供するというプロトコルを採用するから、ホストとクライアントとの間の通信処理が簡単であり、また、ネットワーク等を介してクライアント若しくは情報端末装置にサービスを提供するデータ処理システムの低コスト化も実現できる。

【００４６】

また、ネットを介してプログラムを提供するような情報処理方法、例えば前述のように、インターネット等を介するプログラムのダウンロード、或いは、クライアント内で作られたプログラムにおいて使われるソフト部品をサーバーが提供するような方法を想定する。このとき、クライアントのリソースの観点より全てのプログラムを自分で実行することが困難若しくは不経済な場合がある。そこで、ネットワークを介してサーバーがクライアントにサービスを提供する情報処理方法では、サーバーは、一体として使われるべき第１のプログラムと第２のプログラムの内、第１のプログラムをネットワークを介してクライアントへ送信し、該クライアントからの要求に応答して、前記第２のプログラムを実行する

【００４７】

【発明の実施の形態】

《システム開発方法》

本発明に係るシステム開発方法の具体例を説明する。ここでは、組み込み制御にマイクロコンピュータを用いるターゲットマシンをＣ＋＋高級言語を開発言語として開発する場合について説明する。

【００４８】

図２にはターゲットマシンの開発に用いるデータ処理システムが例示される。ホストマシン１はパーソナルコンピュータ若しくはエンジニアリングワークステーション等の計算機とされ、伝送路若しくはネットワーク２を介してホストマシン１にターゲットマシン３が接続される。ホストマシン１は、そこに組み込まれているプログラム及びデータによって機能が実現され、ターゲットマシンの開発段階においてはコンパイラシステム、デバッガシステムなどの開発環境を構成している。特に説明するまでもないが、ホストマシン１は、プロセッサ、ＲＡＭ、ＲＯＭ及びインタフェースコントローラ等が実装され、インタフェースコントローラにはディスプレイ、キーボード、ハードディスクなどの補助記憶装置が接続され、プロセッサの動作プログラムに従ってデータ処理可能にされる。

【００４９】

ターゲットマシン３の開発に当り、ターゲットマシンに実装されるマイクロコンピュータが実行するプログラムは、先ずフルセットのＣ＋＋高級言語によってソースプログラムとして記述される。

【００５０】

図１にはシステム開発方法で利用される各種ソフトウェアの関係が示される。コンパイラ５はソースプログラム４をデバッガスクリプト６とロードモジュール７に変換する。ロードモジュール７はターゲットマシン３のアプリケーションプログラム８及びデバッグモニタプログラム９としてターゲットマシン３にダウンロードされる。デバッガスクリプト６は、ホストマシン１上でデバッガ１０及び通信機能プログラム１１として実行される。

【００５１】

前記コンパイラ５は、フルセットのＣ＋＋言語で記述されたソースプログラム４をターゲットマシン３のマイクロコンピュータ用のオブジェクトプログラムにコンパイルする機能を有する。このとき、Ｃ＋＋高級言語機能のうち、リソース（メモリ、入出力機能等）の限られたマイクロコンピュータにおいてオーバーヘッドなく実現できる範囲、例えばＪＩＳのＣ言語仕様（ＪＩＳ規格「プログラム言語Ｃ」）における「フリースタンディング仕様」の範囲の記述（第２の記述）

については、前記マイクロコンピュータ上のオブジェクトコード（第３コード若しくはその他のコード）に変換する。ソースプログラムにおけるそれ以外の処理を規定する記述（第１の記述）、即ち例外処理などの高級言語の高機能記述に対しては、ホストマシン１で実行可能な命令若しくはコマンド列（第１コード）に変換し、また、ターゲットマシン３側で実行可能なオブジェクトコードとして、例えば所定のＴＲＡＰ命令のコード（第２コード）に変換する。第１コードは前記第１の記述によって定義される機能をホストマシン１上で実現するコードである。これに対して第２コードは、第１の記述によって定義される機能とは異なる機能をターゲットマシン３上で実現するコードであればよい。要するに、第２コードにはターゲットマシン３に負荷をかけずに実行可能なコードを選べばよい。このように、コンパイラは、高級言語の機能単位に着目してターゲットマシン３に負荷をかける言語機能単位をホストマシン１で実行可能にするものであり、高級言語の機能単位に着目してターゲットマシン３の負荷を軽減させる。

【００５２】

ターゲットマシン３における前記第２コードの実行によってホストマシン１に第１コードを実行させることを想定する。そのためには、デバッガ１０とターゲットマシン３との間での連絡が必要になり、それには以下の最低限の機能があればよい。最低限の機能は、①ターゲットマシン３からデバッガ１０への処理の要求、②デバッガ１０によるターゲットマシン３のメモリへの読み込み及び書き込み、③デバッガ１０によるターゲットマシン３のレジスタへの読み込み及び書き込み、である。これは、ターゲットマシン３のマイクロコンピュータのためのデバッグモニタプログラムによって実現できる機能である。

【００５３】

ここでは、ターゲットマシン側ではそれらの機能①②③を、ターゲットマシン３上のマイクロコンピュータ（ターゲットマイコン）のアプリケーションプログラム内でＴＲＡＰ命令を発行することによって起動される割り込みハンドラの処理を介して支援する。このとき、デバッガ側の機能として必要なのは、各ＴＲＡＰ命令の命令アドレスに対応する第１コードで規定される高級言語機能の実現、次に、ＴＲＡＰ命令の置かれた命令アドレスの情報取得、そして、高級言語機能

をデバッガ側でシミュレートするために必要な、変数とそのターゲットマシン３のメモリ上のアドレスとの対応情報（その時点におけるスタックの状態を示す情報など）の取得である。この点において割り込みハンドラは、ターゲットマシン３上におけるＴＲＡＰ命令コードの命令アドレスを出力する機能を有する。さらに、コンパイラ５は、デバッガ１０側において上記②、③の連絡機能を用いてデバッガ１０側でターゲットマシン３のメモリ及びレジスタの内容を操作可能とするデバッグコマンド列を生成する。デバッグコマンド列及び第１コードは対応するＴＲＡＰ命令の命令アドレスをキーとしてリンクできる対照表若しくは対照テーブルを構成している。

【００５４】

尚、ＴＲＡＰ命令コードは割り込み命令を表すコードであり、ターゲットマシン３におけるマイクロコンピュータのアーキテクチャー上、規定のベクタ等で指し示される割り込みハンドラによる処理に分岐させる命令である。割り込みハンドラはターゲットマシン３のＯＳ（オペレーティングシステム）でサポートしてもよい。或いはコンパイラ５がそのＴＲＡＰ命令コードに対応する規定の割り込みハンドラをそのＴＲＡＰ命令コードにリンクさせてもよい。図１において割り込みハンドラは、デバッグモニタプログラム９の一部として図示されているが、それに限定されない。

【００５５】

図３には前記コンパイラ５による第１の記述に関するコンパイル機能の詳細が例示される。ソースプログラムに含まれる高機能記述部分を抽出すると、その第１記述は、その記述で示される機能をホストシステムで模擬する為に必要なデバッグスクリプト１０，１１に変換され、アプリケーションプログラム８としてのターゲットプログラムのためのＴＲＡＰ命令コード（第２コード）に変換される。デバッグスクリプト１０，１１は前記第１コード及びデバッグコマンド列を含み、対応するＴＲＡＰ命令コードの命令アドレスとの対照表を構成する。

【００５６】

コンパイラ５によりコンパイル処理されたプログラムをデバッグすとき、ホストマシン１はデバッガ１０で動作され、ターゲットマシン３はアプリケーション

プログラム８で動作される。デバッグでは先ず、ターゲットマシン３のプログラムをデバッガ１０を用いて起動する。

【００５７】

ターゲットマシン３が高級言語機能に相当するＴＲＡＰ命令を実行し、これによって、ＴＲＡＰ命令発行時の命令アドレスがデバッガ１０に与えられると、デバッガ１０は、前記対照表によりＴＲＡＰ命令発行時のＰＣ（プログラムカウンタレジスタ、即ちＴＲＡＰ命令自身のアドレス）から、実行すべきデバッグコマンド列と第１コードをサーチして実行する。これにより、デバッガ１０はＴＲＡＰ命令の命令アドレスを用いてその時のターゲットマシン３におけるスタック領域の情報などを取得し、第１コードで示される高級言語機能をデバッガ側で模擬することができる。このように、Ｃ＋＋の例外処理機能をターゲットマシン３に代わってデバッガ１０がシミュレートすることにより、システムリソースの限られたターゲットマシン３をＣ＋＋のフルセット仕様の開発環境を用いて開発することが可能にある。

【００５８】

デバッグが終了し、その結果、ターゲットマシン３上では例外処理機能等の高機能が不用になれば、デバッガモニタプログラム、換言すればＴＲＡＰ命令の割り込みハンドラを、単にＴＲＡＰ命令を無視するソフトウェアと切換えて最終ソフトウェアを構築すればよい。或いは、ＴＲＡＰ命令を埋め込んだ命令アドレスが分かっているから、ＴＲＡＰ命令をＮＯＰ（ノン・オペレーション）命令に置き換えて、最終的なアプリケーションプログラムとしてもよい。このことからも理解されるように、デバッガ１０は、そのようなＴＲＡＰ命令が発生したかどうかを表示できる機能を持つことが望ましい。

【００５９】

ターゲットマシン３を伝送路若しくはネットワーク上で実現する場合には、ターゲットマシン３の最終プログラムにはＴＲＡＰ命令及びデバッグモニタプログラム（割り込みハンドラ）を含め、最終的に開発されたシステムの稼動後も、高級言語機能のオーバーヘッド部分を規定する第１コード及びデバッグコマンド列をホストマシンに残し、ターゲットマシン３が前記所定のＴＲＡＰ命令を実行し

たとき、デバッグ時と同様にその高級言語機能をホストマシンでシミュレートしてやればよい。

【００６０】

このように、オーバヘッドの大きな高級言語の機能を、ターゲットマシン上ではＴＲＡＰ命令１命令で実現できるから、ターゲットマシンのオーバヘッドを最小にでき、リソースの少ないマイクロコンピュータ若しくはターゲットマシン上で高級言語によって開発されたプログラムを効率よく実行できる。

【００６１】

ホストマシンにターゲットマシンのオーバヘッド機能部分を負担させることにより、ターゲットマシンの価格を下げることができる。電話局と携帯電話のように、ホストマシン１個に多数のターゲットマシンが対応するシステムにおいて特に顕著である。ターゲットマシンのリソースが少なくても、高級言語をフルスペックで実装することができる。組込み用途のシングルチップマイクロコンピュータ、そのコンパイラやデバッガ、通信ネットワークで特に効果がある。

【００６２】

《オブジェクト解放処理の模擬》

前記Ｃ＋＋で記述されたターゲットマシンの為の例外処理をホストマシンで模擬するための具体例について説明する。

【００６３】

図４にはＣ＋＋の言語機能を模式的に示してある。図４では、特に制限されないが、Ｃ＋＋言語の全ての機能を、オーバーヘッドの大きな例外処理機能（Ｆ１）、テンプレート・クラス機能（Ｆ２）、Ｃ言語機能（Ｆ３）に分類してある。ここでは、例外処理機能（Ｆ１）をＴＲＡＰ命令とデバッガスクリプトへの変換対象とする。その他の機能は、ターゲットマシンのオブジェクトコードへの変換対象とする。

【００６４】

前記例外処理機能を説明する。Ｃ＋＋言語における例外処理は、Ｃａｔｃｈ構文によって例外処理の有効となる範囲と例外処理時の操作を宣言し、ｔｈｒｏｗ文の実行により例外条件を発生させる。Ｃａｔｃｈ構文とｔｈｒｏｗ文は必ずし

も同じ関数内に無くてもよい。Ｃａｔｃｈ構文及びｔｈｒｏｗ文を用いた記述の簡略化された例は図３のソースプログラムの記述として示されている。

【００６５】

Ｃ＋＋言語における例外処理の問題点は、例外処理が登録される関数と例外を発生する関数に対してオーバヘッドとなる余分な処理が発生するだけでなく、その間で、（動的に）呼び出され得る全ての関数において例外発生の可能性に対して備えなければならないということである。

【００６６】

この点を更に明確化するために、Ｃ＋＋言語における例外発生時の処理を説明する。例外が発生すると、①Ｃａｔｃｈ構文の例外処理の宣言のプログラム点に制御を移行し、②Ｃａｔｃｈ構文のある関数からｔｈｒｏｗ文のある関数までに関数呼び出しによって割り付けられたスタック領域を解放し、③Ｃａｔｃｈ構文で宣言された処理を実行する。

【００６７】

このとき、Ｃ＋＋言語はオブジェクト指向言語であるため、関数呼び出しによって割付けられたスタック領域を開放するとき、当該関数で作成したオブジェクトの解放処理を実施する必要がある。

【００６８】

Ｃ＋＋言語におけるオブジェクトの解放処理を実現する通常行なわれている一方式を図５に基づいて説明する。図５の（Ａ）の関数ｆ、ｇに対して割当てられたスタックフレームの一例が（Ｂ）に示されている。図５のスタックフレームにおける下位アドレスは紙面の下方である。親の関数ｆのスタックフレームの上位側アドレスに子の関数ｇのスタックフレームが形成されている。

【００６９】

〔Ｉ〕各関数のスタックフレームの最下位アドレスには２ワードの領域を確保し、１ワード目には親の関数のフレームの先頭アドレス（フレームポインタ）、２ワード目には当該関数のオブジェクトを開放するための関数のアドレス（解放処理関数のアドレス）を保持する。例えば図５の（Ｂ）に示される関数ｇのスタックフレームでは、１ワード目にはｇのフレームポインタＰ１ｇが配置され、２ワ

ード目にはｇの解放処理関数のアドレスＰ２ｇが配置される。

【００７０】

〔ＩＩ〕例外発生時に当該関数のオブジェクト解放処理実施後、スタックフレームが例外処理を登録した関数に達するまで、〔ＩＩＩ〕のスタック解放処理を実施する。

【００７１】

〔ＩＩＩ〕子関数ｇは解放処理関数のアドレスを呼び出すことにより自フレーム内のオブジェクト（オブジェクト０、オブジェクト１、…）の解放処理を実施し、更にフレームポインタＰ１ｇにより親関数ｆのスタック領域最下位アドレスを求め、ｇのスタックフレームを全て開放する。

【００７２】

しかし、例外処理をターゲットマシン自身が実行しないのであれば、図６に例示されるように、各スタックフレームの上位２ワードのフレームポインタ及び解放処理関数のアドレスのエリア（例えばＰ１ｇ，Ｐ２ｇ）を省くことができる。これによって、貴重なオンチップＲＡＭの領域を節約することができる。即ち、ターゲットマシン若しくはその実装マイクロコンピュータのリソースを節約することができる。換言すれば、マシンリソースに制限がある場合には、前記フレームポインタ及び解放処理関数のアドレスのエリアを省くために、前述のシステム開発方法で説明したようにして、例外処理をターゲットマシン自身で実行させなければよい。

【００７３】

但し、それら２ワードは例外処理を登録する関数や例外処理を発生する関数において必要なだけでなく、これら登録と例外の間に（動的に）呼び出され得る全ての関数に必要である。したがって、Ｃ＋＋言語ではＣ言語に対して全ての関数においてスタックフレーム内に２ワードの余分な領域が必要になる。異なる実現方法を採ったとしても、Ｃ＋＋の例外処理という言語機能が全ての関数に対して影響を与え、Ｃ言語よりもリソースを必要とする機械語を要求することに変わりない。

【００７４】

ターゲットマシン３上のＲＡＭにフレームポインタ及び解放処理関数のアドレスのエリア（例えばＰ１ｇ，Ｐ２ｇ）を持たないで、例外処理時の前記〔Ｉ〕～〔ＩＩＩ〕の操作を、実行するためには、デバッガは以下の処理を実行する。

【００７５】

〔ＩＶ〕ターゲットマシンにおけるＴＲＡＰ命令の命令アドレス（例外が発生したときのプログラムカウンタ）から、ＴＲＡＰ命令を含む関数名と、関数内のプログラムカウンタの値を求める。

【００７６】

〔Ｖ〕当該関数のデバッグ情報から、関数の出口（オブジェクト解放処理の直前）でのスタックポインタと現スタックポインタの値の差を求め（これはコンパイラが管理する情報であり、デバッグ情報に含めることができる）、スタックポインタを関数の出口における値に変更する。

【００７７】

〔ＶＩ〕当該関数のデバッグ情報から、当該関数のオブジェクト解放処理関数のアドレスを求め、その関数を呼び出す。呼び出すためには解放処理関数の出口にブレークポイントを設け、プログラムカウンタの値を解放処理関数のエントリに設定して再実行すればよい。

【００７８】

〔ＶＩＩ〕当該関数にデバッグ情報から関数の出口でのスタックフレームのサイズを求め、親関数が当該関数を呼び出したところまでスタックを解放する。

【００７９】

〔ＶＩＩＩ〕上記〔ＩＶ〕～〔ＶＩＩ〕を、例外処理を登録した関数のスタックポインタの位置に達するまで繰り返す。

【００８０】

〔ＩＸ〕尚、解放処理の中で例外が発生する可能性があるため、上記処理のネストをデバッガが管理する必要がある。

【００８１】

上記〔ＩＶ〕～〔ＩＸ〕の処理は図１で説明した前記第１コードに含まれてい

る。これら処理は、全てデバッガが持つデバッグ情報に基づいて実施するため、ターゲットマシンにおいて関数のスタックフレーム上にこの処理に必要な情報を置く必要はない。要するに、デバッガ１０がターゲットマシン３の例外処理を代替する場合でも、スタックフレームに前記フレームポインタ及び解放処理関数のアドレスを格納する２ワードのエリアを設けることは一切不要になる。実際、〔Ｖ〕、〔ＶＩＩ〕で用いるスタックフレームサイズや関数内でのプログラムカウンタの値に対するスタックポインタのオフセット値が図５の（Ｂ）におけるフレームポインタの役目を果たし、また、〔ＶＩ〕で用いる、当該関数のオブジェクト解放処理のアドレス（デバッグ情報内）が図５の（Ｂ）におけるスタックフレーム内のオブジェクト解放処理のアドレスの役目を果たすため、これら２ワードをターゲット装置のＲＡＭ上のスタックフレーム内に保持する必要は全くない。

【００８２】

《システム開発方法》

次に、伝送路若しくはネットワークを介してサーバーがクライアントにサービスを提供する情報処理システムの観点に立ったシステム開発方法を説明する。

【００８３】

図７にはクライアントのシステムを開発するときのシステム開発方法が例示される。クライアントのソースプログラムはフルセットのＣ＋＋言語で記述される。システム開発装置はクライアントに実装されるマイクロコンピュータをサポートするコンパイラ及びデバッガを組み込んだパーソナルコンピュータ若しくはエンジニアリングワークステーション等の計算機が用いられる。

【００８４】

システム開発装置はソースプログラムを入力する（Ｓ１）。入力されたソースプログラムの記述に対してサーバーで実行する記述であるか否かを判定する（Ｓ２）。例えば前述のＪＩＳのＣ言語仕様（ＪＩＳ規格「プログラム言語Ｃ」）における「フリースタンディング仕様」の範囲の記述についてはクライアントが実行する記述（第２の記述）と判定し、ソースプログラムにおけるそれ以外の処理を規定する記述、即ち例外処理などの高級言語の高機能記述に対しては、サーバーが実行する記述（第１の記述）と判定する。或いは、高級言語の高機能記述に

加えて、前記「フリースタンディング仕様」の範囲の記述であても処理時間が長いと予め考えられる処理、利用頻度が少ないと予め想定される処理例えば故障診断或いは自己救済機能に関する記述についても第１の記述と判定する。

【００８５】

第２の記述に対しては、クライアント用のオブジェクトコード（第３コード）に変換する（Ｓ３）。前者の第１の記述に対しては、先ずその記述をクライアント用の前記ＴＲＡＰ命令コード（第２コード）に変換し（Ｓ４）、クライアントのメモリ空間における当該ＴＲＡＰ命令の命令アドレスを保持し（Ｓ５）、更に第１の記述が示す機能をサーバー用のオブジェクトコード（第１コード）変換し、クライアント内のメモリやレジスタをリード・ライトする為のデバッグコマンド列に相当するクライアントアクセス用コマンド列を生成する（Ｓ６）。最後に、Ｓ３及びＳ４で生成されたクライアント用オブジェクトプログラム２０、Ｓ５乃至Ｓ７で生成されたサーバ用のオブジェクトプログラム及びクライアントアクセス用コマンド列２１が出力される（Ｓ８）。前記サーバー用オブジェクトプログラム及びクライアントアクセス用コマンド列２１は、クライアント用オブジェクトプログラム２０に含まれるＴＲＡＰ命令コードの命令アドレスとリンクされた対照テーブルを構成する。

【００８６】

このように、システム開発装置は、前述の図１の説明と同様に、クライアントに負荷をかけると考えられる処理をサーバーで実行可能にするものであり、クライアントのマシンリソースを考慮してその負荷を軽減させる。

【００８７】

特にこの例では、クライアントの負荷を最小限とするために、ＴＲＡＰ命令による割り込みハンドラはクライアントの専用ＯＳがサポートし、クライアントによるＴＲＡＰ命令の実行により、当該ＴＲＡＰ命令のアドレスを出力し、その後、サーバーからのリード・ライト要求に答える通信制御を支援するようになっている。

【００８８】

図８には図７の方法で生成されるプログラムが例示される。図８においてソー

スプログラム２２は例えばＰ１、Ｐ２、Ｐ３の記述を含んでいる。図７の方法で出力されるクライアント用のオブジェクトプログラム２０はＰＯ１（Ａ１）、ＴＲＡＰ（Ａ２）、ＰＯ３（Ａ３）のオブジェクトコードを含む。Ａ１，Ａ２，Ａ３はコードのマッピングアドレスを意味する。すなわち、クライアント内のマイクロコンピュータが管理するメモリ空間において、オブジェクトコードが配置されるところのアドレスを表している。命令コードＰＯ１，ＰＯ３はソース記述Ｐ１，Ｐ３が変換されたものである。図７の方法で出力されるサーバー用オブジェクトプログラム及びクライアントアクセス用コマンド列２１は、ソース記述Ｐ２をサーバーのオブジェクトコードに対応させて変換したプログラムＰＯ２とクライアントアクセス用コマンド列をＴＲＡＰ命令のアドレスＡ２でリンクさせたテーブル構造を有する。

【００８９】

図９には前記クライアントのシステムを開発するときのシステム開発装置３０が示される。クライアントのソースプログラムはフルセットのＣ＋＋言語で記述される。システム開発装置は前記クライアントに実装されるマイクロコンピュータをサポートするコンパイラ及びデバッガを組み込んだパーソナルコンピュータ若しくはエンジニアリングワークステーション等の計算機３０が用いられる。コンパイラ及びデバッガ等の開発支援プログラムは単数又は複数毎のフロッピディスク（ＦＤ）３１やＣＤ－ＲＯＭディスク３２等の記憶媒体に原始的に記録されて提供され、それら記録媒体に記録されたプログラムは計算機内部のハードディスク装置３３などの記録媒体にインストールされて保持される。或いは、そのようなプログラムは、公衆回線網３４などを介してダウンロードされ、前記ハードディスク装置３３に記録されている。前記開発支援プログラムは、図７に示したステップを図９の計算機３０に実行させる。

【００９０】

尚、図１乃至図６で説明したシステム開発方法に利用する前記コンパイラ５等のシステム開発プログラムもＣＤ－ＲＯＭ３２などの記録媒体に記録されてシステム開発装置に提供される。この時のシステム開発装置は図２ではホストマシン１であり、ホストマシン１はその記録媒体からシステム開発プログラムを読み込

んで実行する。

【００９１】

《情報処理システム》

次に、伝送路若しくはネットワークを介してサーバーがクライアントにサービスを提供する情報処理システムを説明する。

【００９２】

図１０には情報処理システムの一例が示される。情報処理システムは、ネットワーク４４のサーバー４０と複数のクライアント４１～４３が接続されて成る。ネットワーク４４は、公衆回線網、携帯電話網、ＬＡＮ（ローカルエリアネットワーク）、インタネット、或いはそれらの複数種類混在であってもよい。サーバー４０はワークステーションやパーソナルコンピュータなどの計算機を有し、代表して記憶ユニット４０Ｓが例示される。クライアント４１～４３は携帯電話機、ＰＤＡなどの携帯情報端装置等であって、ＣＰＵのような計算機を有し、代表してメモリ４１Ｍ～４３Ｍが示される。記憶ユニット４０Ｓ、メモリ４１Ｍ～４３Ｍはデータ及びプログラム格納エリアとして利用される。

【００９３】

クライアント４１～４３の動作プログラムは図７のシステム開発方法で得られたクライアント用オブジェクトプログラム２０とされる。このクライアント用オブジェクトプログラム２０は図９のシステム開発装置３０から夫々のメモリ４１Ｍ～４３Ｍに格納される。格納形態はクライアント４１～４３の製造段階で画一的に行ない、或いはネットワーク４４にシステム開発装置３０が接続されている場合にはクライアントが個別的にダウンロードで行なうことができる。格納先は個々のクライアント４１～４３のメモリ４１Ｍ～４３Ｍである。前述の如く、クライアントにとってオーバーヘッドの大きな処理はＴＲＡＰ命令に置き換えてサーバー４０に処理させるので、そのためのサーバー用オブジェクトプログラム及びクライアントアクセス用コマンド列を含む対照テーブル２１は前記システム開発装置３０からサーバー４０の記憶ユニット４０Ｓに格納される。この格納形態もクライアントと同様に製造段階或いはネットワーク経由の何れでもよい。

【００９４】

図１１にはクライアント４１の具体例が示される。クライアント４１は、ＣＰＵ（中央処理装置）５０、入出力ユニット５１、及び通信ユニット５２を有する。ＣＰＵ５０はフェッチした命令を解読して制御信号を生成する命令制御部と、その制御信号で制御される演算部を有し、フェッチした命令を実行する。図１１では演算部のレジスタ４１Ｒ及びメモリ４１Ｍが代表的に例示される。メモリ４１ＭはＣＰＵの動作プログラム等を保有する例えば電気的に書き換え可能な不揮発性メモリ、及びＣＰＵのワーク領域とされるＲＡＭを備える。入出力ユニット５１はディスプレイ、キースイッチ、スピーカ、マイクロフォンなどを有し、マンマシンインタフェース機能を実現する。通信ユニットはクライアント４１を外部とインタフェースする手段であり、例えば、携帯電話機であれば、高周波部、変復調処理部、Ａ／Ｄ・Ｄ／Ａ変換部などを有し、携帯電話網を介して通信可能にされる。或いは、ＬＡＮアダプタ、ＭＯＤＥＭアダプタ、シリアルインタフェース、パラレルインタフェースなどであってもよい。その他のクライアント４２，４３も同様に構成される。

【００９５】

図１２には前記サーバー４０の具体例が示される。サーバー４０は、ワークステーションなどの計算機５３、通信ユニット５４、及び前記記憶ユニット４０Ｓを有する。計算機５３は例えば実行ユニットとしてマイクロプロセッサ５３Ｐ及びマイクロプロセッサ５３Ｐのワーク領域として利用されるＲＡＭ５３Ｍを有する。計算機５３はその他に図示を省略するマンマシンインタフェース回路などを備えている。記憶ユニット４０Ｓはサーバー用プログラム領域と共にクライアント用プログラム領域を有する。クライアント用プログラムは、前記サーバー用オブジェクトプログラム及びクライアントアクセス用コマンド列を含む対照テーブル２１を格納する領域である。サーバーとしての固有の動作の為の動作プログラムはサーバー用プログラム領域に格納される。通信ユニット５４はネットワーク４４に接続可能なルータ、モデム、ＬＡＮ等の通信機能を有する。

【００９６】

図１３にはクライアント４１による前記クライアント用オブジェクトプログラ

ム２０の実行制御状態が示される。

【００９７】

前記クライアント４１は、前記ＴＲＡＰ命令コードを含むクライアント用オブジェクトプログラム２０をメモリ４１Ｍに保有し、ＣＰＵ５０がそのクライアント用オブジェクトプログラム中のＴＲＡＰ命令コードを実行すると、対応する割り込みハンドラが起動され、当該命令コードのアドレス（ＣＰＵ５０が管理するメモリ空間におけるアドレス）を通信ユニット５２を介して外部へ送信する（Ｓ１０）。前記アドレスの外部送信に応答してサーバー４０から、その時のスタック領域の状態などの出力要求があると、これを入力し（Ｓ１）、入力した要求に応答して、スタック領域の状態等の内部情報をアクセスしてサーバー４０に提供する（Ｓ１２）。ＴＲＡＰ命令のような命令を実行するだけで、クライアント４１がそのプログラムメモリ空間上のどの命令実行状態にあるかをサーバー４０に連絡でき、サーバー４０は、そのアドレスに対応する処理を行なうことによって、クライアント４１にとってオーバヘッドの大きな処理を代わって実行することができる。

【００９８】

図１４にはサーバー４０による前記対照テーブル２１を用いたプログラム及びコマンド実行制御状態が示される。

【００９９】

サーバー４０の計算機５３は、ＣＰＵ５０を含むクライアント４１から当該クライアントで実行されるプログラム中の所定の命令コード例えばＴＲＡＰ命令コードのアドレス情報（アドレスＰＣ）を受け取り、これによって、前記対照テーブル２１のプログラム実行要求のあることを判定する（Ｓ２０）。計算機５３は、受け取ったアドレス情報をキーとして対照テーブル２１をサーチし、そのアドレス情報にリンクされているサーバー用オブジェクトプログラム及びクライアントアクセス用コマンド列を記憶ユニット４０ＳからＲＡＭ５３Ｍにロードする（Ｓ２１）。計算機５３はクライアントアクセス用コマンド列を実行し、そのアドレス情報に基づいてその時の前記クライアント４１のスタック領域の状態等の内部情報を取り込み（Ｓ２２）、取り込んだ内部情報を用いて、そのアドレス情報

にリンクされている対照テーブル２１のサーバー用オブジェクトプログラムを実行する（Ｓ２３）。このサーバー４０は、クライアント４１におけるオーバヘッド機能を代替することができる。サーバー４０は、前記内部情報を取り込むために、前記取り込んだアドレス情報に基づいて、取り込むべき前記内部情報のアドレスを求める処理を、前記対照テーブル２１のクライアントアクセス用コマンド列を実行して生成するので、この点においてもクライアント４１に負担をかけない。

【０１００】

《情報処理方法》

図１０の情報処理システムにおいてサービスの要求をネットワーク４４を介して受けることを想定したとき、クライアント４１，４２，４３からサーバー４０のプログラムを呼び出すのではなく、サーバー４０はクライアント４１，４２，４３の命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供することになる。即ち、ネットワーク４４を介してサーバー４０がクライアントにサービスを提供する情報処理方法として、サーバー４０は、プログラム（クライアントが実行するプログラム）を格納したクライアント４１，４２，４３から出力されるところの前記プログラムのアドレス空間におけるアドレス情報をネットワークを介して受信し、受信したアドレス情報に対応するサービスの要求に応答するための処理を実行ユニット５３Ｐ，５３Ｍで実行し、実行結果に基づくサービスをネットワーク４４を介してクライアントに送信する。このように、サーバー４０はクライアント４１，４２，４３の命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供するというプロトコルを採用するから、サーバー４０とクライアント４１，４２，４３との間の通信処理が簡単である。更に、クライアント４１，４２，４３において前記アドレス情報を出力する処理を規定するプログラムと、サーバ４０においてアドレス情報に応じた処理を行なうプログラムとを、前述のシステム開発方法を適用して形成することができるからネットワーク４４を介してクライアントにサービスを提供するデータ処理システムの低コスト化も実現できる。

【０１０１】

なお、サーバー４０の受信したアドレス情報がどのクライアントのものであるかは、特に制限されないが、前記受信アドレス情報から識別することを要しない。即ち、クライアント４１，４２，４３とサーバー４０の１対１の通信確立は元々システムが備えている機能を用いればよい。例えばＬＡＮであればクライアントに割当てられるＩＰアドレスを用いて識別すればよいし、携帯電話網であればクライアント固有のＩＤ情報などを用いて識別すればよい。流用可能な識別手段を備えていない場合には前記アドレス情報にクライアント固有のプリフィックスコードを付加してサーバーに与えるようにすれば、当該プリフィックスコードによってサービスを提供すべきクライアントを識別できる。

【０１０２】

また、図１０の情報処理システムにおいて、ネットワーク４４を介してプログラムを提供するような情報処理システムを想定する。即ち、インターネット等を介してプログラムのダウンロード、或いは、クライアント内で作られたプログラムにおいて使われるソフト部品をサーバーが提供するようなシステムを想定する。このとき、クライアントのシステムリソースの観点より全てのプログラムを自分で実行することが困難若しくは不経済な場合がある。これに着目して、図１５に例示されるように、クライアント４１，４２，４３が欲するプログラムを、少なくとも２つ、例えば第１プログラムＰＧＭ１と第２プログラムＰＧＭ２に分ける。サーバー４０は分けられたプログラムを記憶ユニット４０Ｓに保有する。サーバー４０は、一方の第１プログラムＰＧＭ１をダウンロード要求のあるクライアント４１、４２又は４３に提供し、クライアントからの要求で他方の第２プログラムＰＧＭ２を実行する。この実行は、要求を出したクライアントが必要とする結果もしくはサービスを得る為に行なわれるものである。したがって、その実行結果をクライアントに返す必要がある場合は、即座にその結果はクライアントにサービスの提供として返される。例えば第２のプログラムがクライアントの故障診断プログラムである場合には、診断結果に応じたサービスを提供するとよい。ソフトウェア的な故障である場合はプログラムの書換え若しくは再ダウンロードを可能にする。修理を要するハードウェア故障の場合には必要な処置若しくは

連絡先等をアナウンスする。

【０１０３】

前記第１プログラムＰＧＭ１と前記第２プログラムＰＧＭ２は、本来一つ若しくは一体として利用される性質上、共通のソースプログラムより形成されるのが効率的である。この場合には図７等で説明したシステム開発方法を適用すればよく、第１のプログラムＰＧＭ１はクライアント用オブジェクトプログラム２０として位置付けられるプログラムであって、第２のプログラムＰＧＭ２は対照テーブル２１のサーバー用オブジェクトプログラム及びクライアントアクセス用コマンド列として位置付けられるプログラムであってよい。このとき、前記要求は、当然前記第１のプログラムにおけるアドレス情報を主体に行なわれることになる。

【０１０４】

そのような処理形態は図１６に示されるデータ処理方法の一つとして把握する事ができる。即ち、クライアント４１，４２，４３がプログラムの提供をサーバー４０に要求すると（Ｓ３０）、サーバー４０は、クライアント用プログラムをネットワーク４４を介して要求したクライアントに提供する（Ｓ３１）。例えば一体として使われるべき第１のプログラムＰＧＭ１と第２のプログラムＰＧＭ２の内、第１のプログラムをネットワーク４４を介して要求したクライアントへ送信する。その後、サーバー４０は、クライアントからのアドレス供給に応答してプログラムに関するサポートを提供する（Ｓ３２）。このサポートは、例えば前記第２のプログラムＰＧＭ２の実行である。サポートの内容はサーバ側のプログラムの実行だけに限らず、プログラムのバージョンアップ、破壊されたプログラムモジュールの修復等であってよい。その場合にはクライアントが保有するプログラムのバージョンを知る必要があり、前記スタック領域の情報取得と同様に所定のクライアントアクセスコマンド列を実行してバージョン情報をクライアントから取得し、そのバージョンに応じたサポートを行なえばよい。

【０１０５】

以上本発明者によってなされた発明を実施形態に基づいて具体的に説明したが、本発明はそれに限定されるものではなく、その要旨を逸脱しない範囲において

種々変更可能であることは言うまでもない。

【０１０６】

例えば、Ｃ＋＋を高級言語の一例として説明した、Ｆｏｒｔｒａｎ９０、Ｊａｖａなどの場合も同様に適用することができる。要するにクライアント若しくはターゲット装置にとってオーバーヘッド機能と考えられる機能をＴＲＡＰ命令のようなコードとそのコードのアドレス出力等によってサーバー若しくはホスト装置に代替させればよい。また、プログラムの記憶媒体はＣＤ－ＲＯＭ、ＦＤ、ハードディスクに限定されず、ＭＯ（Magnet-Optics：磁気光学）ディスク、フラッシュメモリカード等、その他の記憶形式を有する媒体であってよい。また、クライアントが出力するアドレスは、ＴＲＡＰ命令のようなコードのアドレスに限定されない。例えばＴＲＡＰ命令のアドレスではなく、そのＴＲＡＰ命令に対して数命令前に実行された命令コードのアドレスであってもよい。要するに、クライアントから供給されたアドレスに基づいて、サーバーがクライアント内の情報（代替を要求された処理を、サーバーが実現するために使う情報）へのアクセスをできるように、サーバーに代替を要求する処理を特定できるアドレスであって、クライアント内のＣＰＵによって管理されるアドレスであればよい。

【０１０７】

【発明の効果】

本願において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記の通りである。

【０１０８】

Ｃ＋＋言語の例外処理機能に代表されるようなクライアント若しくはターゲット装置にとってオーバーヘッド機能と考えられる機能を、ＴＲＡＰ命令のような命令コードに置き換え、この命令コードの実行により例えばそのコードのアドレスを出力してサーバー若しくはホスト装置にオーバーヘッド機能の実行を代替させるから、ターゲット装置にとってオーバーヘッドになり得る高級言語機能も含めてフルセットの高級言語機能を利用可能にするシステム開発方法を実現することができる。これは、システムリソースの制限を受け易い組込みシステムにおいて、フルセットのＣ＋＋言語等の高級言語を用いたシステムを開発を可能にし、

組込みシステムに関しても高級言語によるプログラムの標準化が可能になり、ワークステーション上で高級言語を用いて開発されたプログラムの移植も可能になる。

【０１０９】

前記システムの開発方法を実現するプログラムを記録した記憶媒体の提供によって前記システム開発方法によるシステム開発を容易化することができる。

【０１１０】

サービスの要求を伝送路若しくはその集合体であるネットを介して受け、そのサービスを提供するホスト装置を有するような情報処理システムを想定したとき、情報端末装置からホスト装置のプログラムを呼び出すのではなく、ホスト装置は情報端末装置の命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供するというシステム構成を採用することにより、ホスト装置と情報端末装置との間の通信処理を簡素化でき、また、ネットワーク等を介して情報端末装置にサービスを提供するデータ処理システムのの低コスト化、データ処理方法の効率化を実現できる。

【図面の簡単な説明】

【図１】

本発明に係るシステム開発方法で利用される各種ソフトウェアの関係を例示する説明図である。

【図２】

ターゲットマシンの開発に用いるデータ処理システムを例示するシステム構成図である。

【図３】

コンパイラによる第１の記述に関するコンパイル機能の詳細を例示する説明図である。

【図４】

Ｃ＋＋の言語機能を模式的に示した説明図である。

【図５】

Ｃ＋＋言語におけるオブジェクトの解放処理を実現するためにスタックフレー

ムが通常備えるフレームポインタ及び解放処理関数アドレスの領域を示す説明図である。

【図６】

フレームポインタ及び解放処理関数アドレスの領域を省いたスタックフレームの説明図である。

【図７】

クライアントのシステムを開発するときのシステム開発方法を例示するフローチャートである。

【図８】

図７の方法で生成されるプログラムを例示する説明図である。

【図９】

クライアントのシステムを開発するときのシステム開発装置を例示するシステム構成図である。

【図１０】

ネットワークを介してサーバーがクライアントにサービスを提供する情報処理システムを例示するブロック図である。

【図１１】

クライアントの具体例を示すブロック図である。

【図１２】

サーバーの具体例を示すブロック図である。

【図１３】

クライアントによるクライアント用オブジェクトプログラムの実行制御状態を例示するフローチャートである。

【図１４】

サーバによるサーバ用オブジェクトプログラム及びクライアントアクセス用コマンド列の実行制御状態を例示するフローチャートである。

【図１５】

図１０の情報処理システムにおいてネットワークを介してプログラムを提供するような情報処理システムを想定したときのサーバーの機能を概略的に示したブ

ロック図である。

【図１６】

ネットワーク上でプログラムのアドレス情報を用いてサービスを提供するデータ処理方法の一例を示すフローチャートである。

【符号の説明】

１　ホストマシン

３　ターゲットマシン

４　ソースプログラム

５　コンパイラ

６　デバッガスクリプト

７　ロードモジュール

８　アプリケーションプログラム

９　デバッグも似たプログラム

１０　デバッガ

１１　通信機能プログラム

Ｐ１ｇ　関数ｇのフレームポインタ

Ｐ２ｇ　関数ｇの解放処理関数のアドレス

２０　ソースプログラム

２０　クライアント用オブジェクトプログラム

２１　サーバ用オブジェクトプログラム及びクライアントアクセスコマンド列（対照テーブル）

３０　システム開発装置

３１，３２，３４　記憶媒体

３４　公衆回線

４０　サーバー

４０Ｓ　記憶ユニット

４１，４２，４３　クライアント

４１Ｍ，４２Ｍ，４３Ｍ　メモリ

４４　ネットワーク

５０　ＣＰＵ

５３　計算機

５３Ｐ　プロセッサ

５３Ｍ　ＲＡＭ

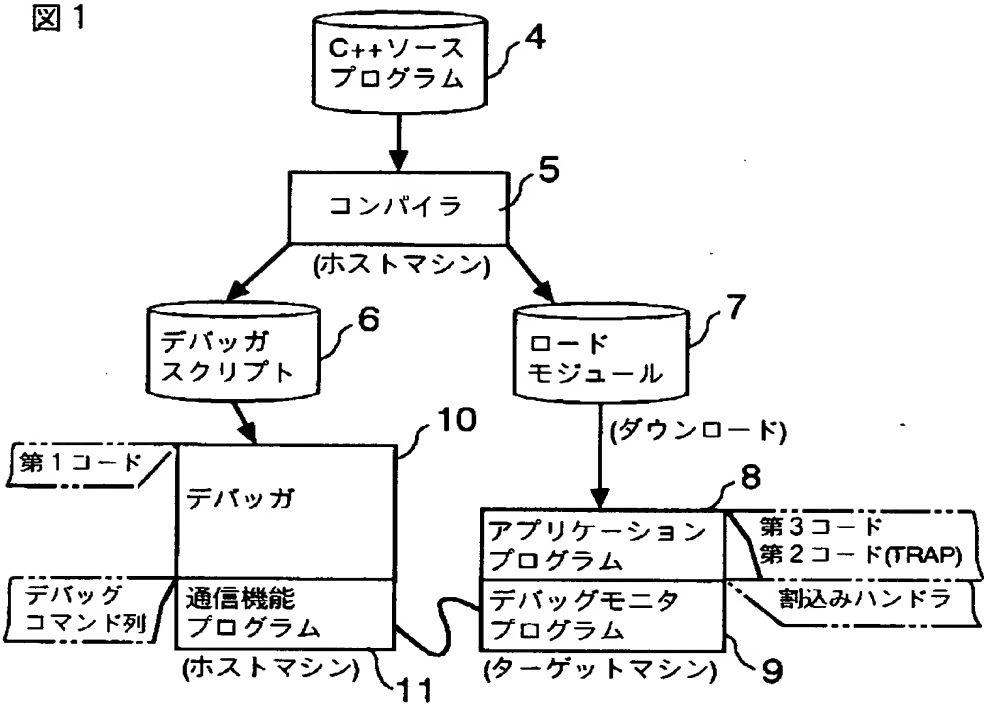
ＰＧＭ１　第１のプログラム

ＰＧＭ２　第２のプログラム

【書類名】　図面

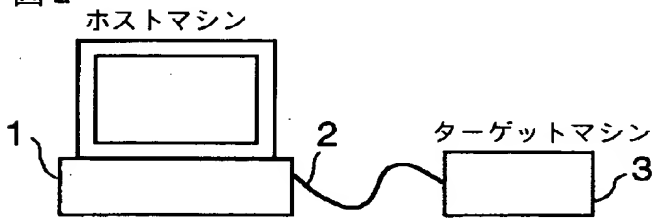
【図１】

図１



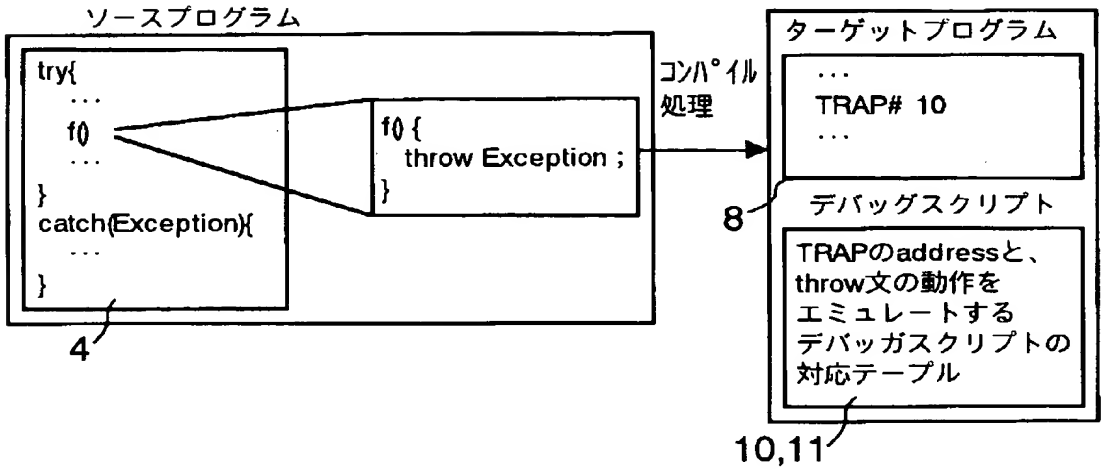
【図２】

図２



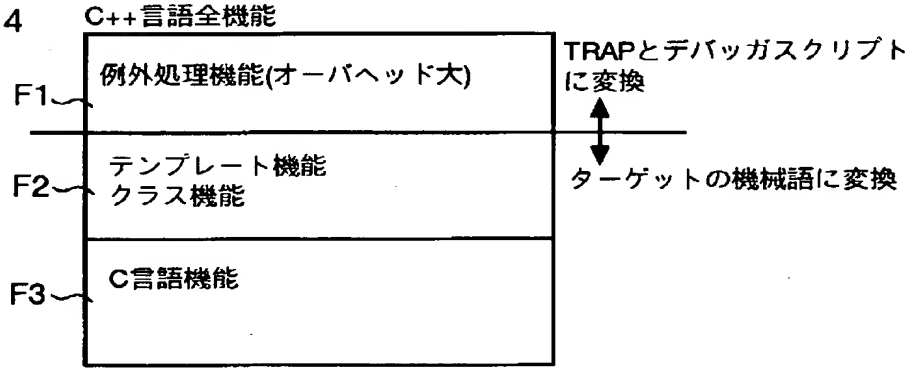
【図３】

図３


ソースプログラム
try{
...
f0
...
}
catch(Exception){
...
}
f0 {
throw Exception ;
}
4
コンパイル
処理
ターゲットプログラム
...
TRAP# 10
...
8
デバッグスクリプト
TRAPのaddressと、
throw文の動作を
エミュレートする
デバッガスクリプトの
対応テーブル
10,11

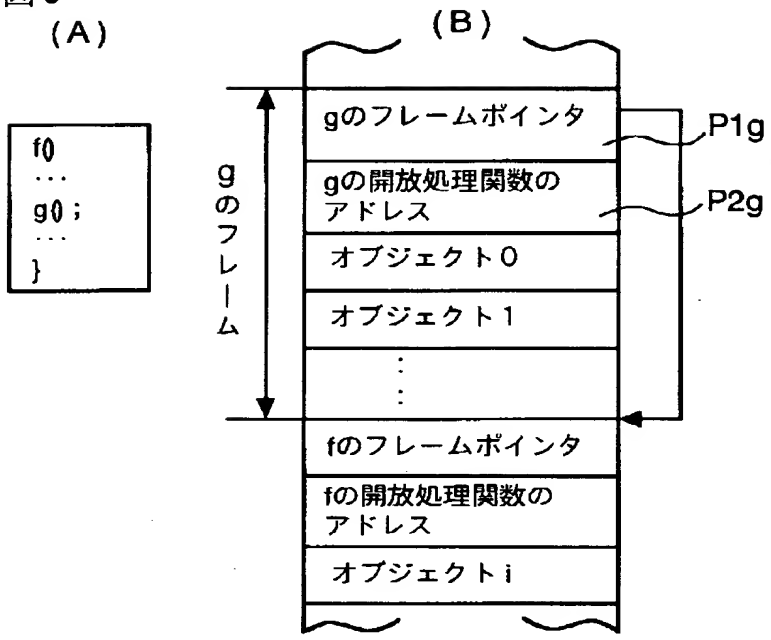
【図４】

図４


C++言語全機能
例外処理機能(オーバヘッド大)
F1
テンプレート機能
クラス機能
F2
C言語機能
F3
TRAPとデバッガスクリプト
に変換
ターゲットの機械語に変換

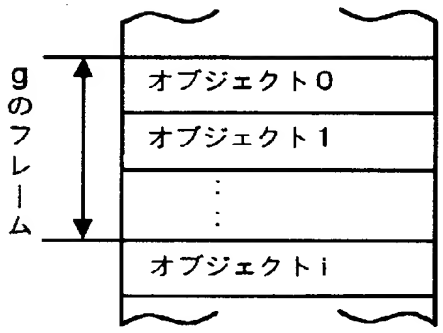
【図５】

図５


(A)
f0
...
g0;
...
}
(B)
gのフレーム
gのフレームポインタ
gの開放処理関数の
アドレス
オブジェクト０
オブジェクト１
fのフレームポインタ
fの開放処理関数の
アドレス
オブジェクトｉ
P1g
P2g

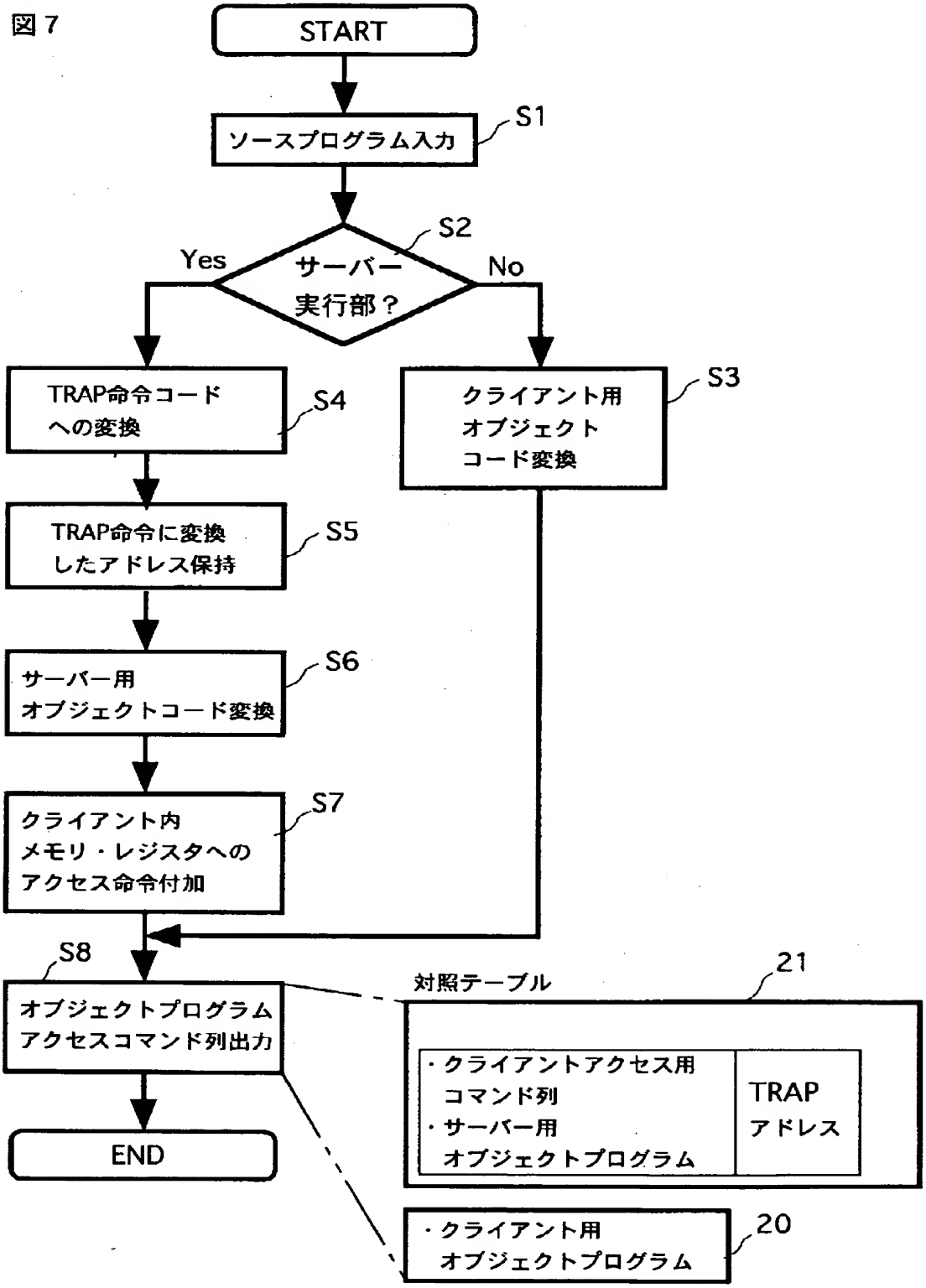
【図６】

図６


gのフレーム
オブジェクト０
オブジェクト１
オブジェクトｉ

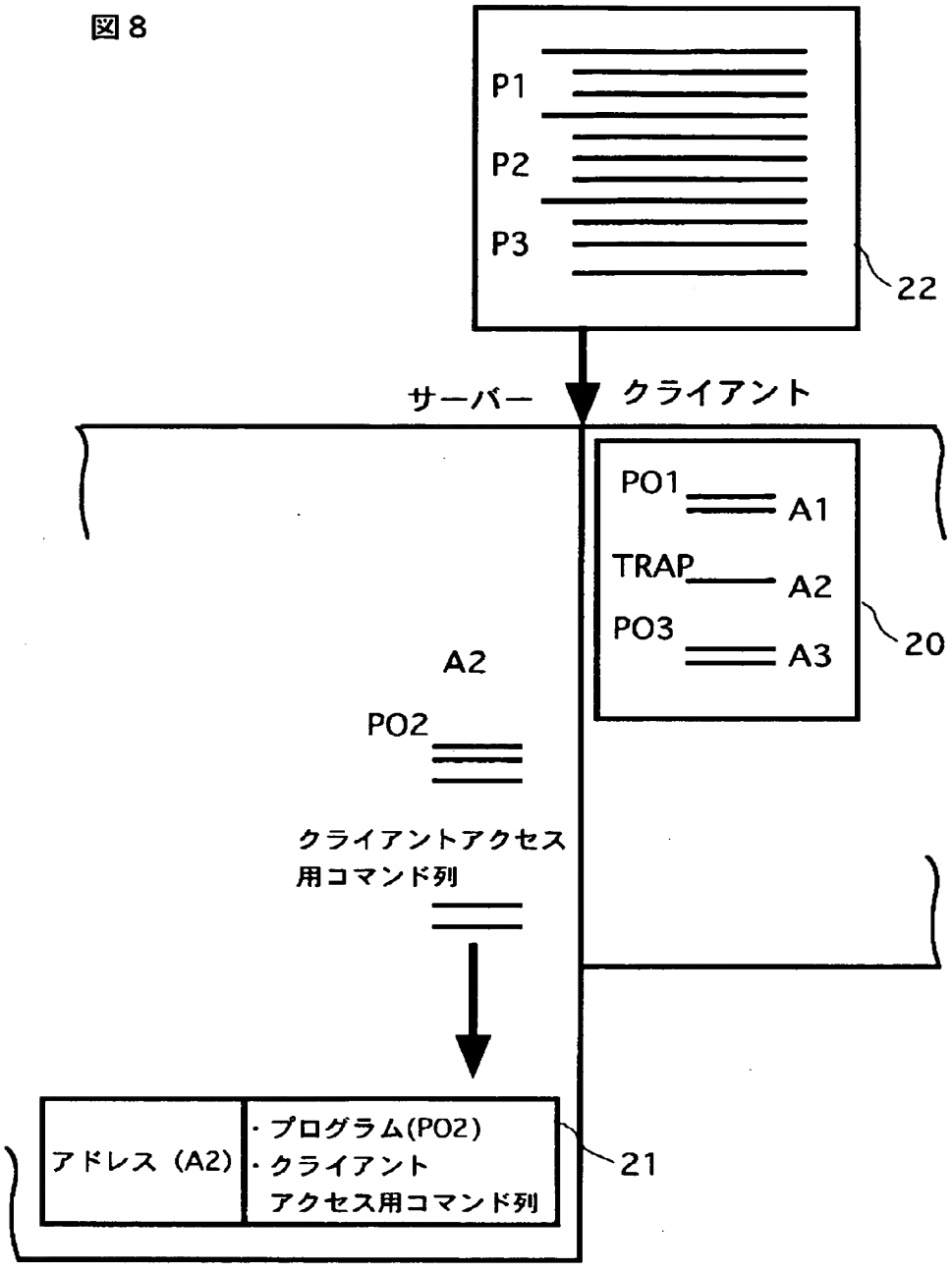
【図７】

図７


START
ソースプログラム入力
S1
S2
サーバー実行部？
Yes
No
TRAP命令コードへの変換
S4
クライアント用オブジェクトコード変換
S3
TRAP命令に変換したアドレス保持
S5
サーバー用オブジェクトコード変換
S6
クライアント内メモリ・レジスタへのアクセス命令付加
S7
S8
オブジェクトプログラムアクセスコマンド列出力
END
対照テーブル
21
・クライアントアクセス用コマンド列
・サーバー用オブジェクトプログラム
TRAPアドレス
・クライアント用オブジェクトプログラム
20

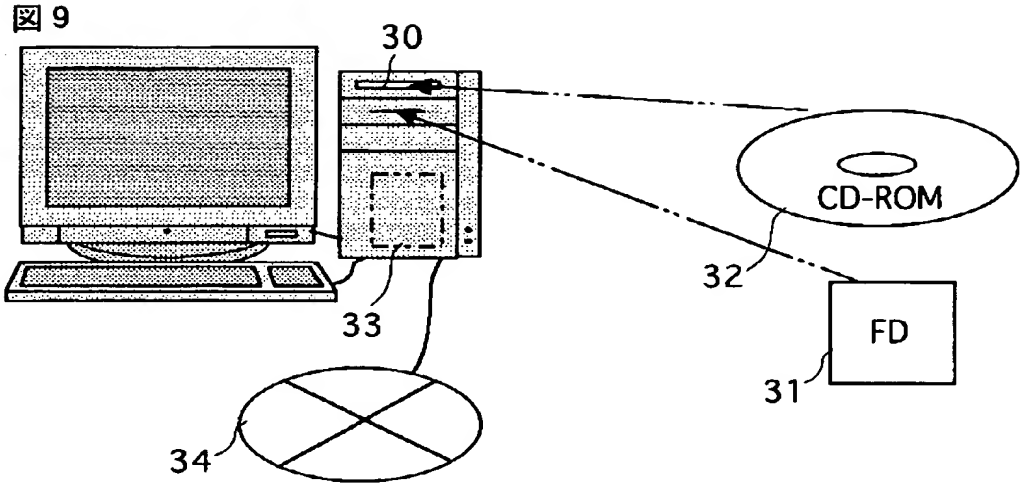
【図８】

図８


P1
P2
P3
22
サーバー
クライアント
PO1
A1
TRAP
A2
PO3
A3
20
A2
PO2
クライアントアクセス
用コマンド列
アドレス（A2）
・プログラム(PO2)
・クライアント
アクセス用コマンド列
21

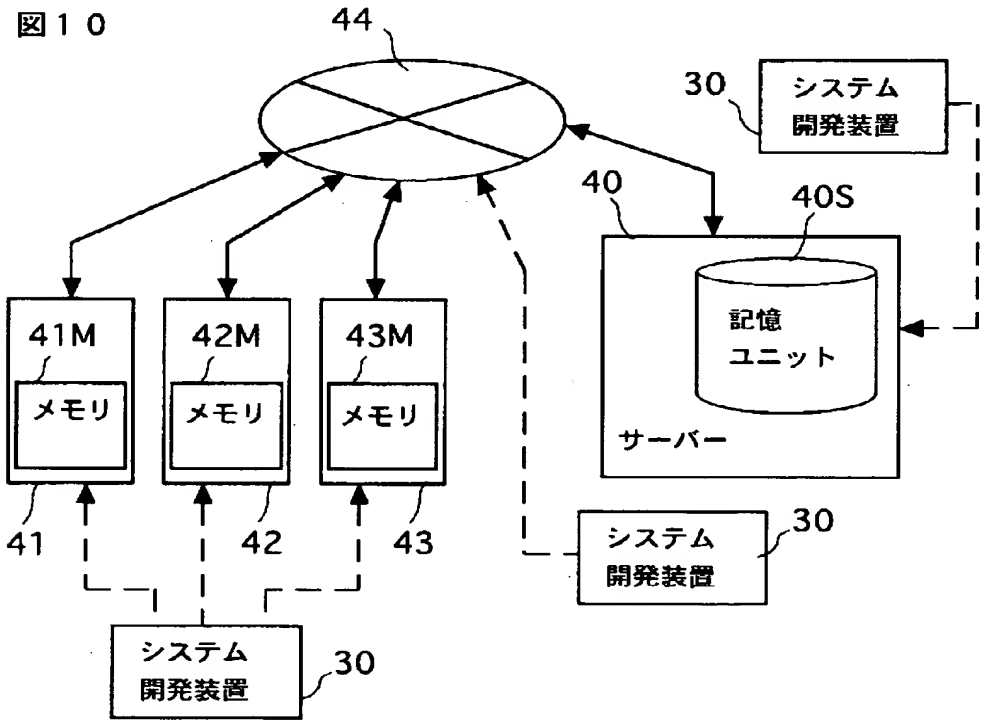
【図９】

図９

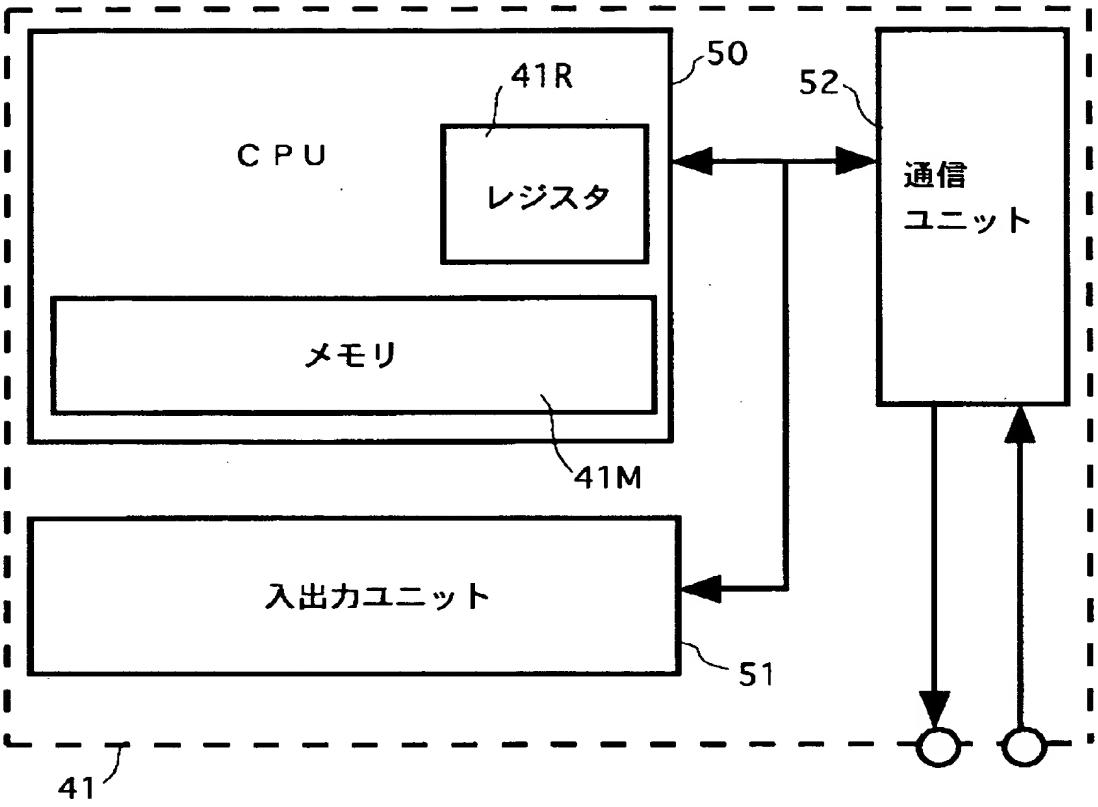

30
CD-ROM
32
FD
31
33
34

【図１０】

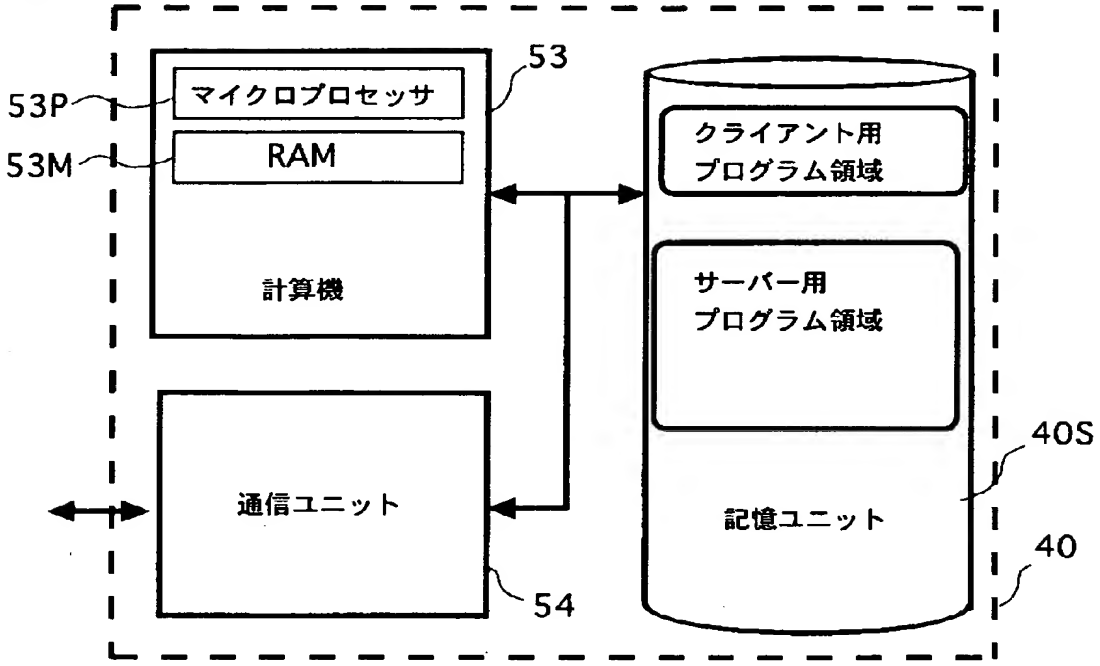
図１０


44
30
システム
開発装置
40
40S
記憶
ユニット
サーバー
41M
42M
43M
メモリ
メモリ
メモリ
41
42
43
システム
開発装置
30
システム
開発装置
30

【図１１】
図１１

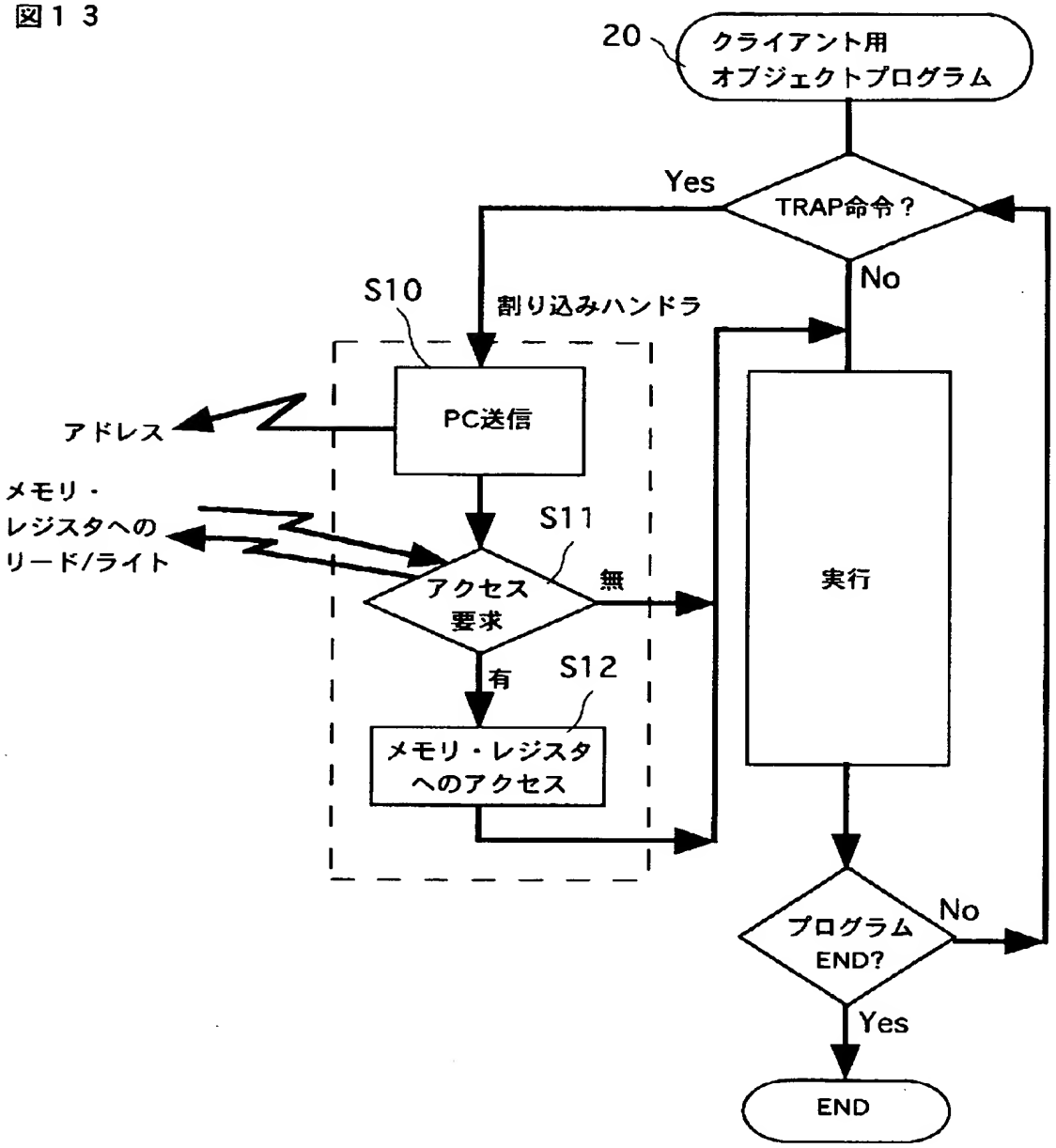


【図１２】
図１２


53
53P
マイクロプロセッサ
53M
RAM
計算機
クライアント用
プログラム領域
サーバー用
プログラム領域
40S
通信ユニット
記憶ユニット
40
54

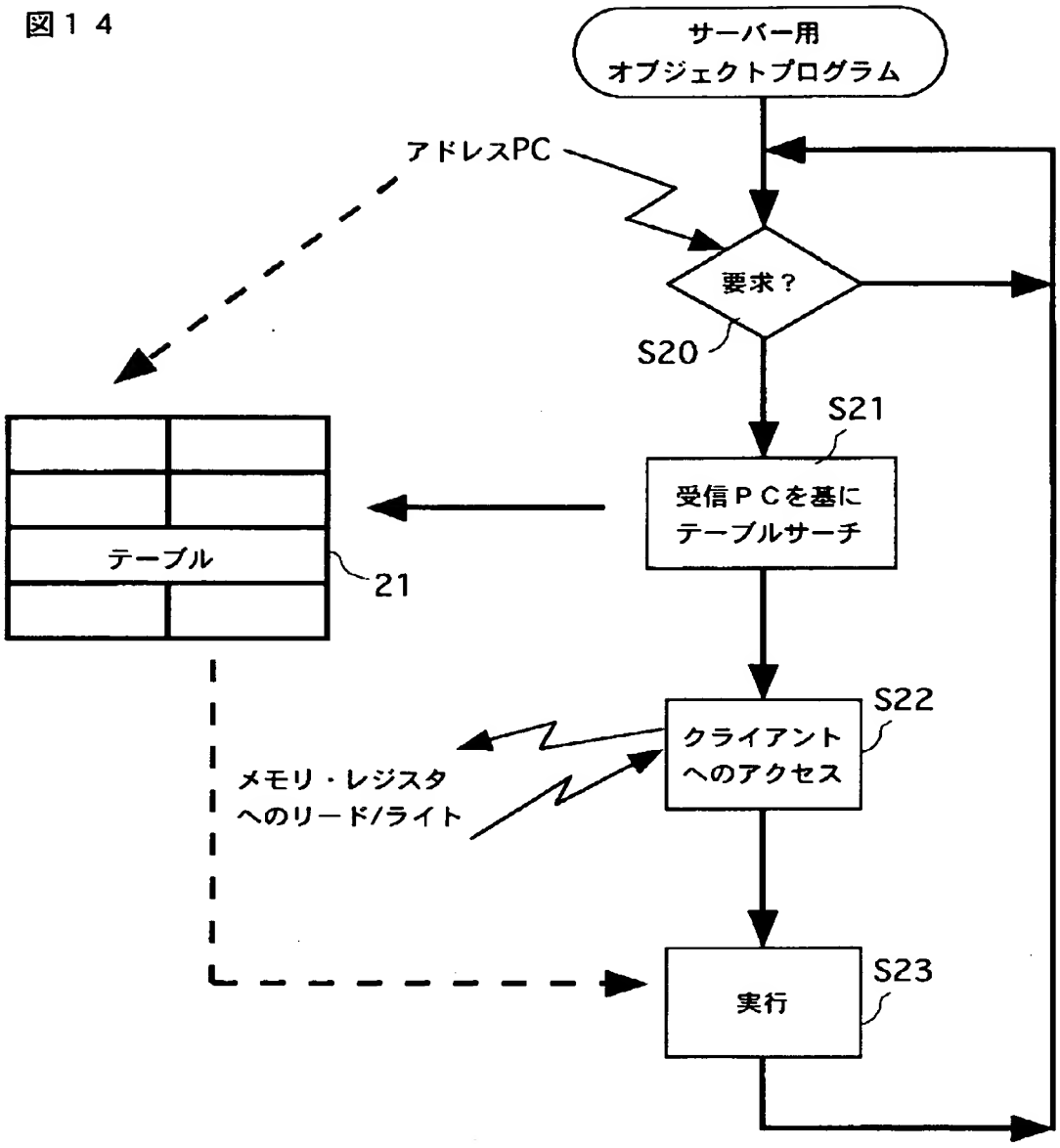
【図１３】

図１３


20
クライアント用
オブジェクトプログラム
Yes
TRAP命令？
No
S10
割り込みハンドラ
PC送信
アドレス
メモリ・
レジスタへの
リード/ライト
S11
アクセス
要求
無
有
S12
メモリ・レジスタ
へのアクセス
実行
プログラム
END?
No
Yes
END

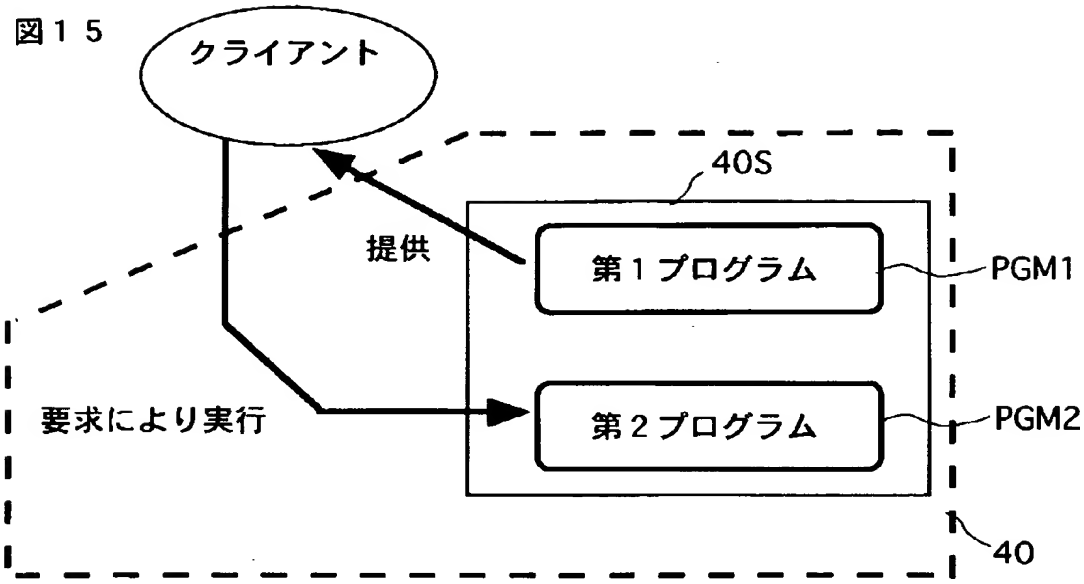
【図１４】

図１４


サーバー用
オブジェクトプログラム
アドレスPC
要求？
S20
S21
受信ＰＣを基に
テーブルサーチ
テーブル
21
S22
クライアント
へのアクセス
メモリ・レジスタ
へのリード/ライト
S23
実行

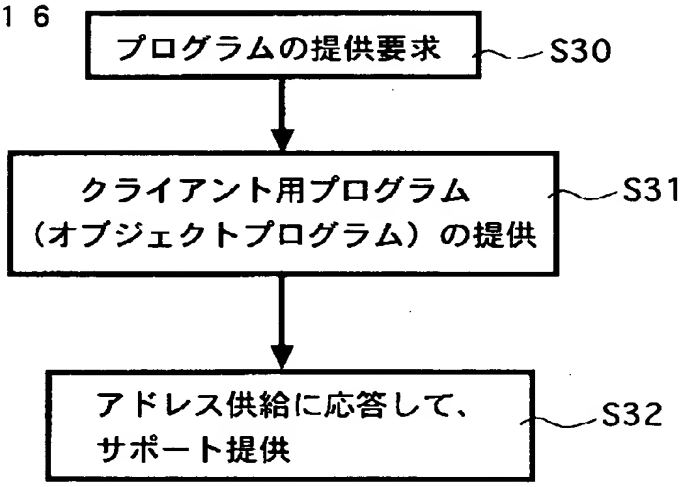
【図１５】

図１５



【図１６】

図１６



【書類名】　　要約書

【要約】

【課題】　ネットワーク等を介してクライアント若しくは情報端末装置にサービスを提供するデータ処理システムの低コスト化、及びクライアントとの間の通信の簡素化を実現する。

【解決手段】　サービスの要求を伝送路若しくはネット（４４）を介して受け、そのサービスを提供するサーバー（４０）を有するような情報処理システムを想定したとき、クライアント（４１，４２，４３）からサーバーのプログラムを呼び出すのではなく、サーバーはクライアントからそのアドレス空間の命令アドレスのようなアドレス情報を受けることによりそのアドレス情報に応ずるサービスを提供する。このプロトコルを採用するから、サーバーとクライアントとの間の通信処理が簡素であり、ネットワーク等を介してクライアント若しくは情報端末装置にサービスを提供するデータ処理システムの低コスト化も実現できる。

【選択図】　　図１０

出　願　人　履　歴　情　報

識別番号　　〔０００００５１０８〕

１．変更年月日　　１９９０年　８月３１日
［変更理由］　新規登録
住　所　東京都千代田区神田駿河台４丁目６番地
氏　名　株式会社日立製作所